Model-based Programming as Estimating, Planning and Executing based on Hidden State

Brian C. Williams

Artificial Intelligence and Space Systems Labs

Massachusetts Institute of Technology

IS Program Review

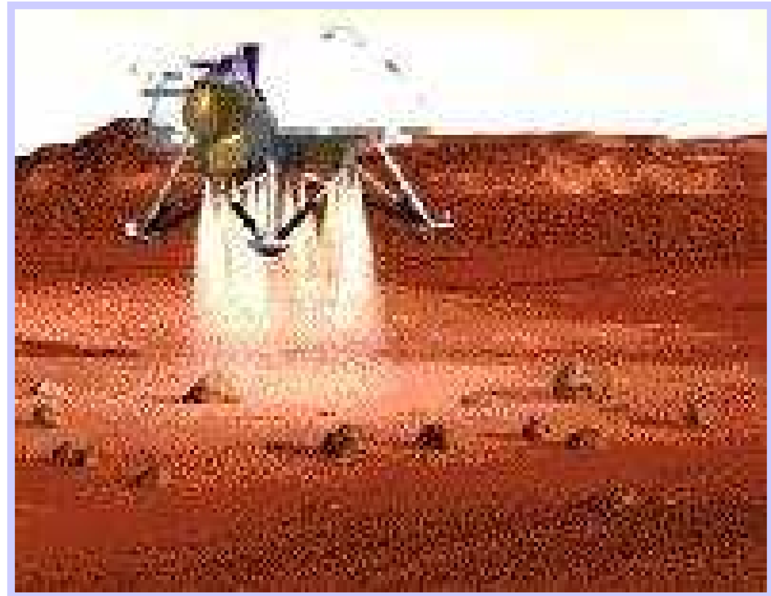
September 5th, 2002

Why Model-based Programming?

Polar Lander Leading Diagnosis:

- Legs deployed during descent.
- Noise spike on leg sensors latched by software monitors.
- Laser altimeter registers 50ft.
- Begins polling leg monitors to determine touch down.
- Latched noise spike read as touchdown.
- Engine shutdown at ~50ft.

Programmers often make commonsense mistakes when reasoning about hidden state.



Objective: Support programmers with embedded languages that avoid these mistakes, by reasoning about hidden state automatically.

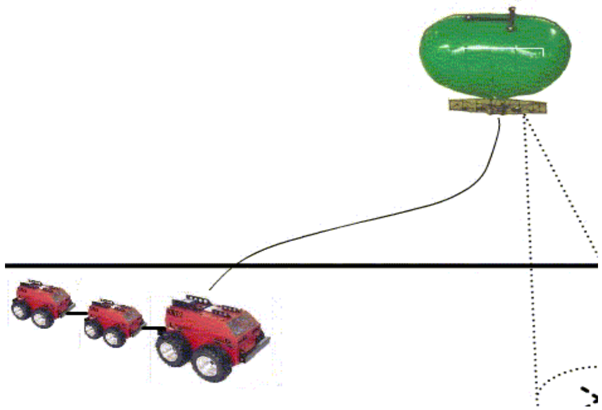
Reactive Model-based Programming

Objective

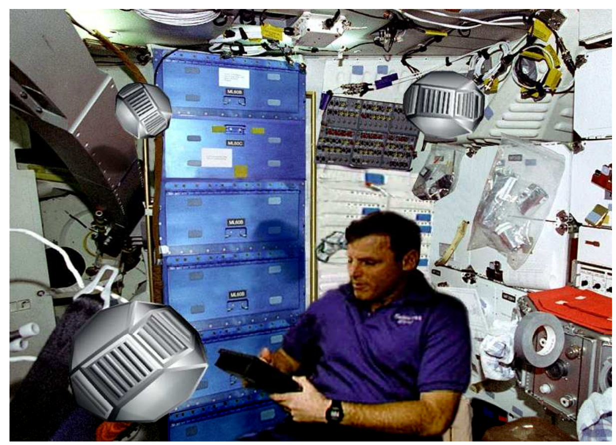
Develop model-based embedded programming languages that think from commonsense models in order to robustly estimate, plan, schedule, command, monitor, diagnose and repair collections of robotic explorers.

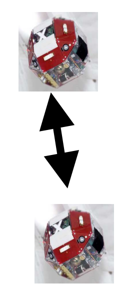
- Reactive Model-based Programming Language
- Titan Model-based Executive

DEMONSTRATION:



Mars 09 Mobile Science Lab



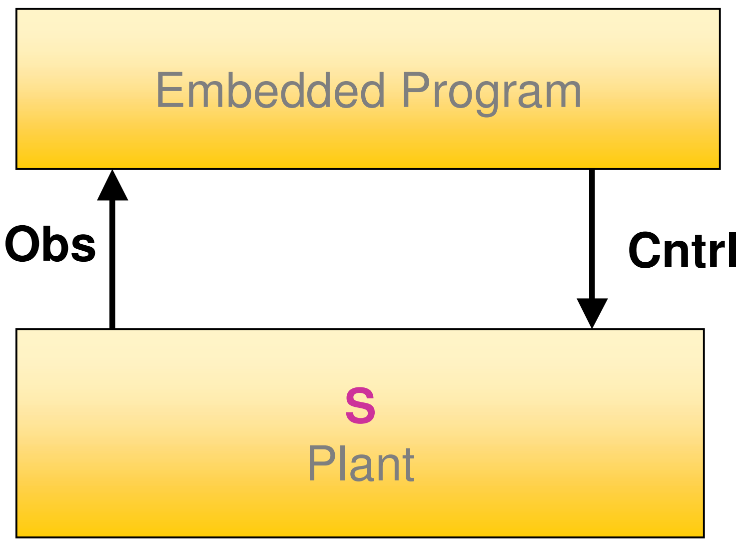


Spheres on ISS (DARPA Funded)
•Robust Station keeping (SIM)
•Robust Docking (MSR)

At the Engineering level, Model-based Programs Interact Directly with State

Embedded programs interact with plant sensors and actuators:

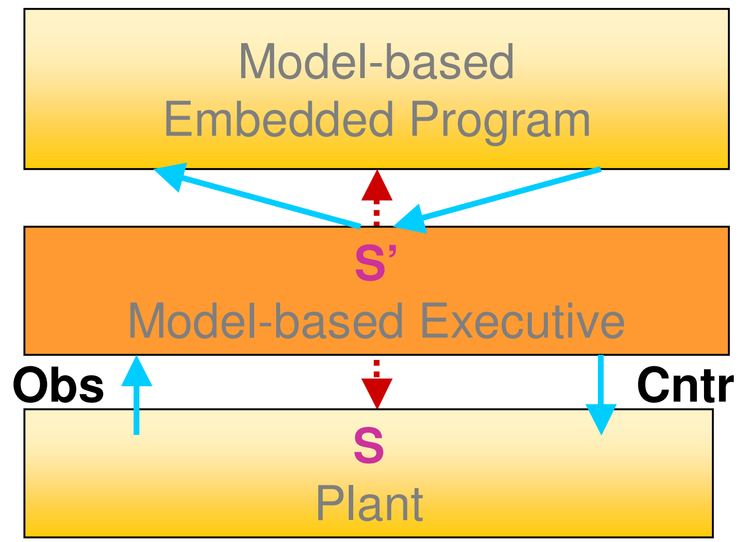
- Read sensors
- Set actuators



Programmers must map between states and sensors/actuators.

Model-based programs interact with plant state:

- Read state
- Write state



Model-based executives map automatically between states and sensors/actuators.

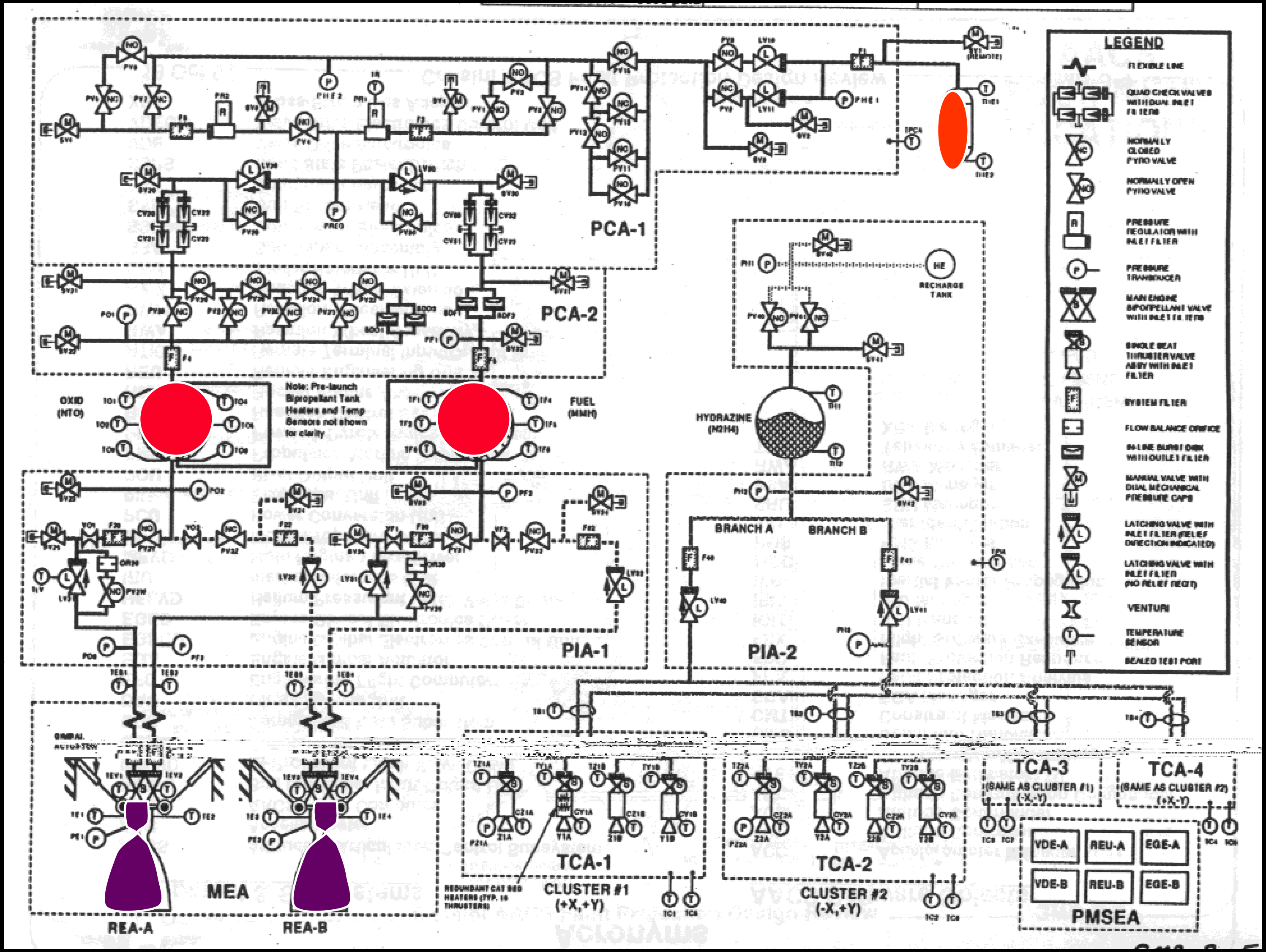
Model-based Executives should automate **ALL** reasoning about system interactions.

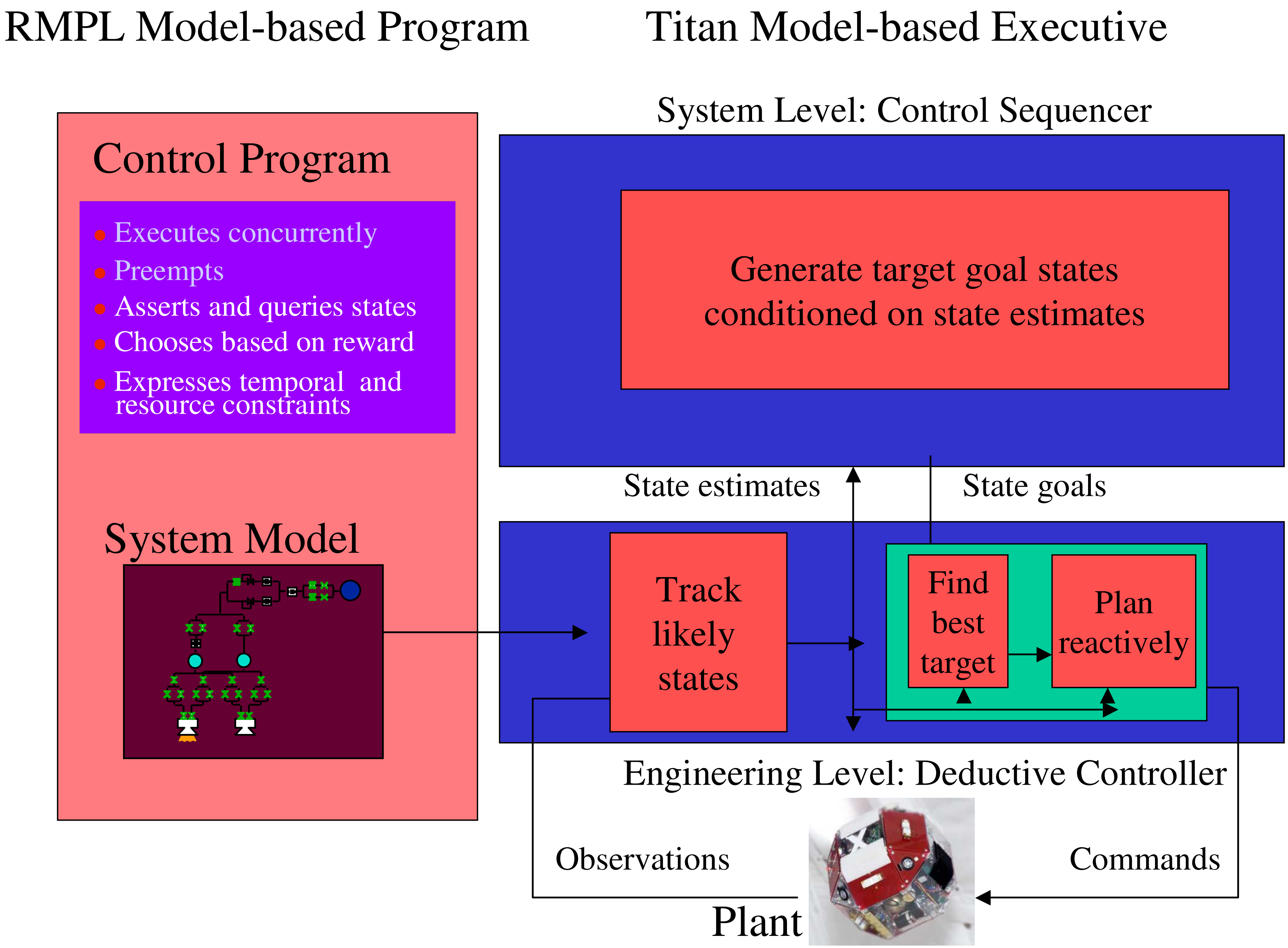
Engineering level:

- Command confirmation
- Diagnosis
- Commanding
- Configuration
- Repair

System level:

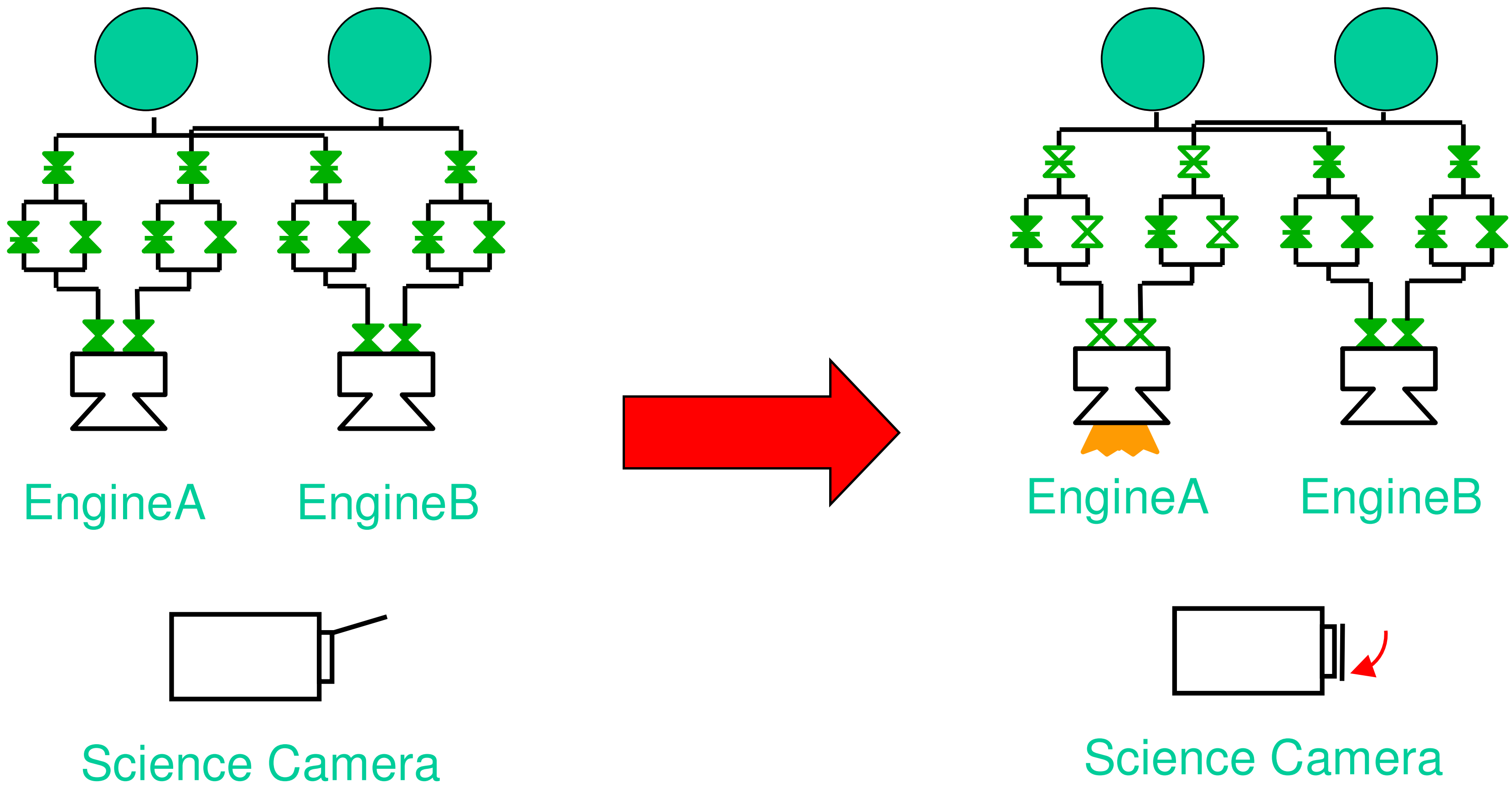
- Generation of contingencies.
- Scheduling




RMPL Model-based Program
Titan Model-based Executive
Control Program
Executes concurrently
Preempts
Asserts and queries states
Chooses based on reward
Expresses temporal and resource constraints
System Model
System Level: Control Sequencer
Generate target goal states conditioned on state estimates
State estimates
State goals
Track likely states
Find best target
Plan reactively
Engineering Level: Deductive Controller
Observations
Commands
Plant

Orbital Insertion Example

Turn camera off and engine on



Control Program

Control program specifies state trajectories:

- fires one of two engines
- sets both engines to 'standby'
- prior to firing engine, camera must be turned off to avoid plume contamination
- in case of primary engine failure, fire backup engine instead

```
OrbitInsert()::

(do-watching ((EngineA = Firing) OR
              (EngineB = Firing))
   (parallel
      (EngineA = Standby)
      (EngineB = Standby)
      (Camera = Off)
      (do-watching (EngineA = Failed)
         (when-donext ( (EngineA = Standby) AND
                        (Camera = Off) )
            (EngineA = Firing)))
      (when-donext ( (EngineA = Failed) AND
                     (EngineB = Standby) AND
                     (Camera = Off) )
         (EngineB = Firing))))
```

Hidden State

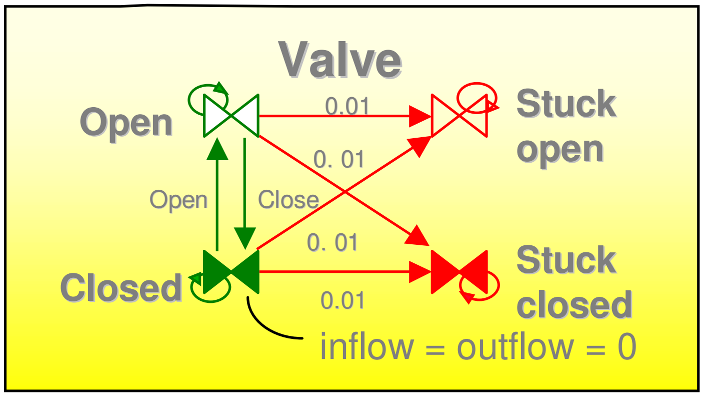
- States like *(EngineA = Standby)* are not DIRECTLY observable or controllable…

Given observations…	*(thrust = zero) AND (power_in = nominal)*
and command history…	last command issued = "standby-cmd"
executive infers "hidden state"	$\Rightarrow$ (EngineA = Standby)
Given state goals executive infers "commands"	*[Turn on DriverA]; [Open ValveA]*

- Thinking in terms of "hidden states" abstracts away complexity of robustly observing and controlling state.
- Model-based executive raises assurance of software by correctly inferring and controlling states.

Synthesize Actions from Models of Complex Behavior

Intended Behavior of System

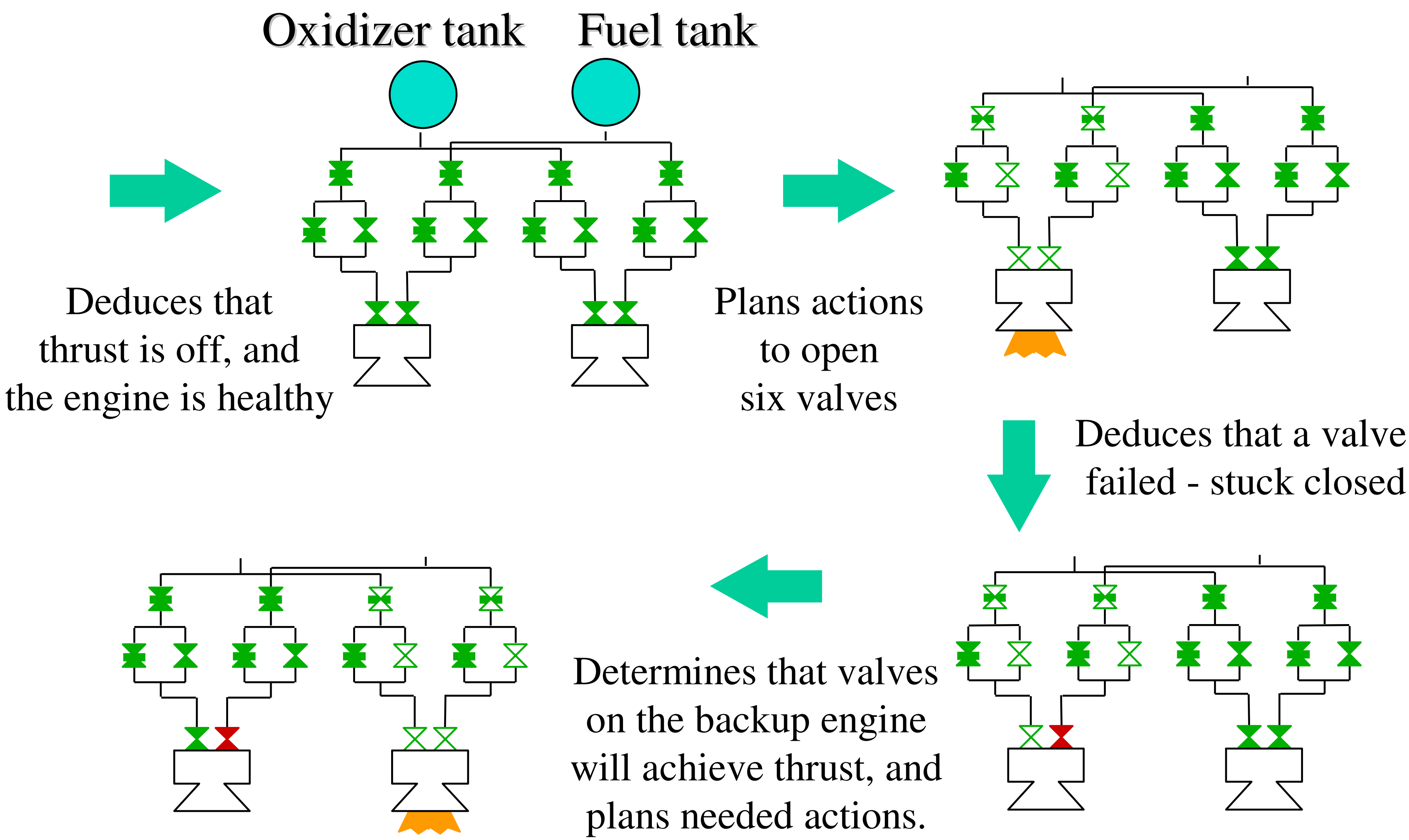


Possible Behaviors of Components

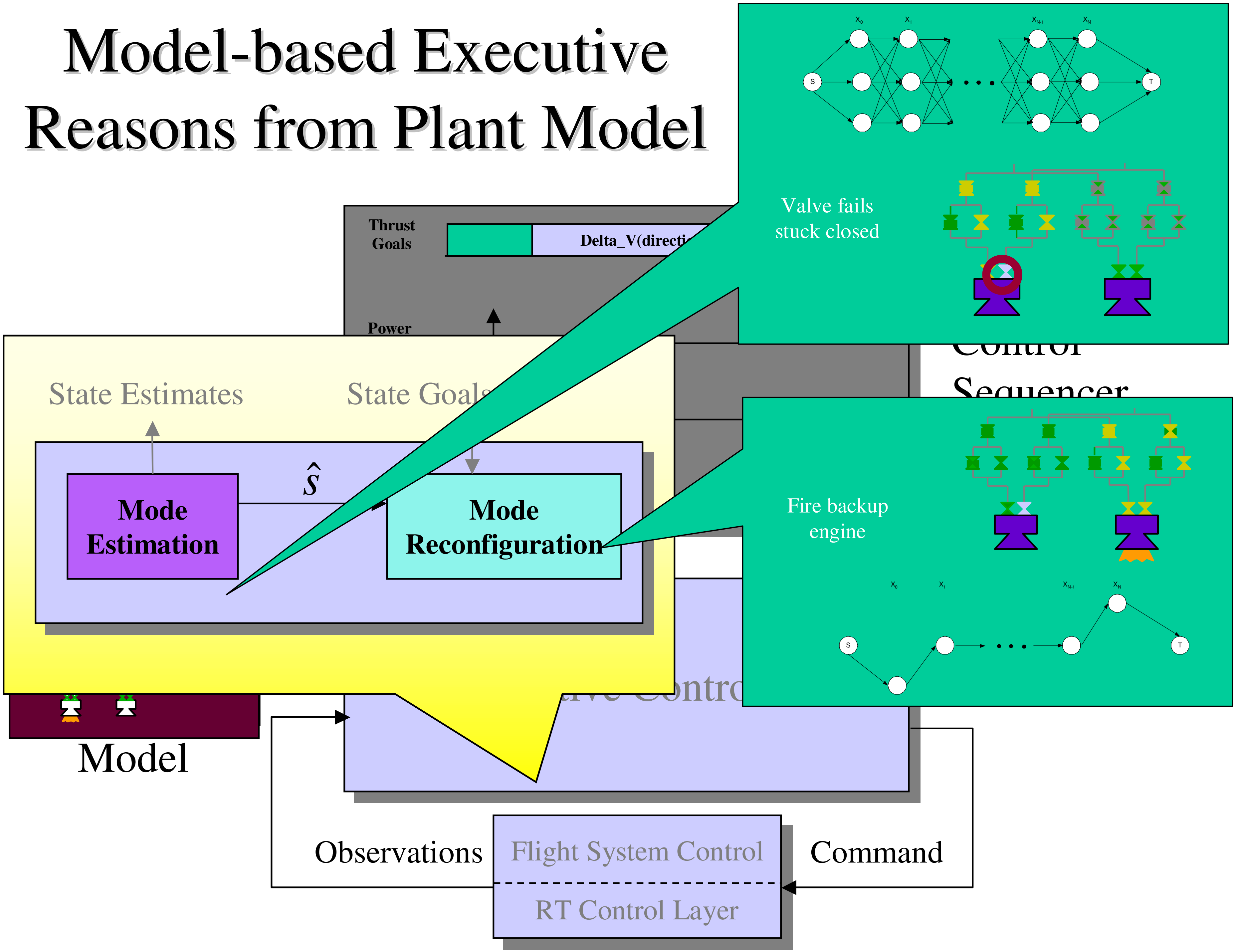
Probabilistic Hierarchical Constraint Automata:

- Complex, discrete and qualitative behaviors
  - modeled through concurrency, hierarchy and non-determinism.
- Anomalies and uncertainty
  - modeled by probabilistic transitions
- Physical interactions
  - modeled by discrete and continuous constraints
- Timing
  - modeled by simple temporal networks

Example: The model-based program sets the state to thrusting, and the deductive controller . . . .



Model-based Executive Reasons from Plant Model



Recent Publications

Model-based Programming:

- B. C. Williams and M. Ingham, "Model-based Programming: Controlling Embedded Systems by Reasoning about Hidden State," *to appear* International Conference on Constraint Programming, September 2002.

MBP & Titan Executive 1. 0:

- B. C. Williams, M. Ingham, S. Chung and P. Elliott, "Model-based Programming of Intelligent Embedded Systems and Robotic Explorers," *to appear* Special Issue on Embedded Software, IEEE Proceedings.

Results: Analysis of Livingstone Deductive Algorithms

Issues:

- Would not explore complete diagnosis space.
- Would not maintain proper ranking of diagnoses in terms of posterior probability.
- Would not rule out all inconsistent diagnoses.

OPSAT:

- Extract Deductive core for solving Optimal Constraint Satisfaction problems.
- Extend to achieve optimality, completeness, and correctness.
- Empirically validate on randomized algorithms and extend

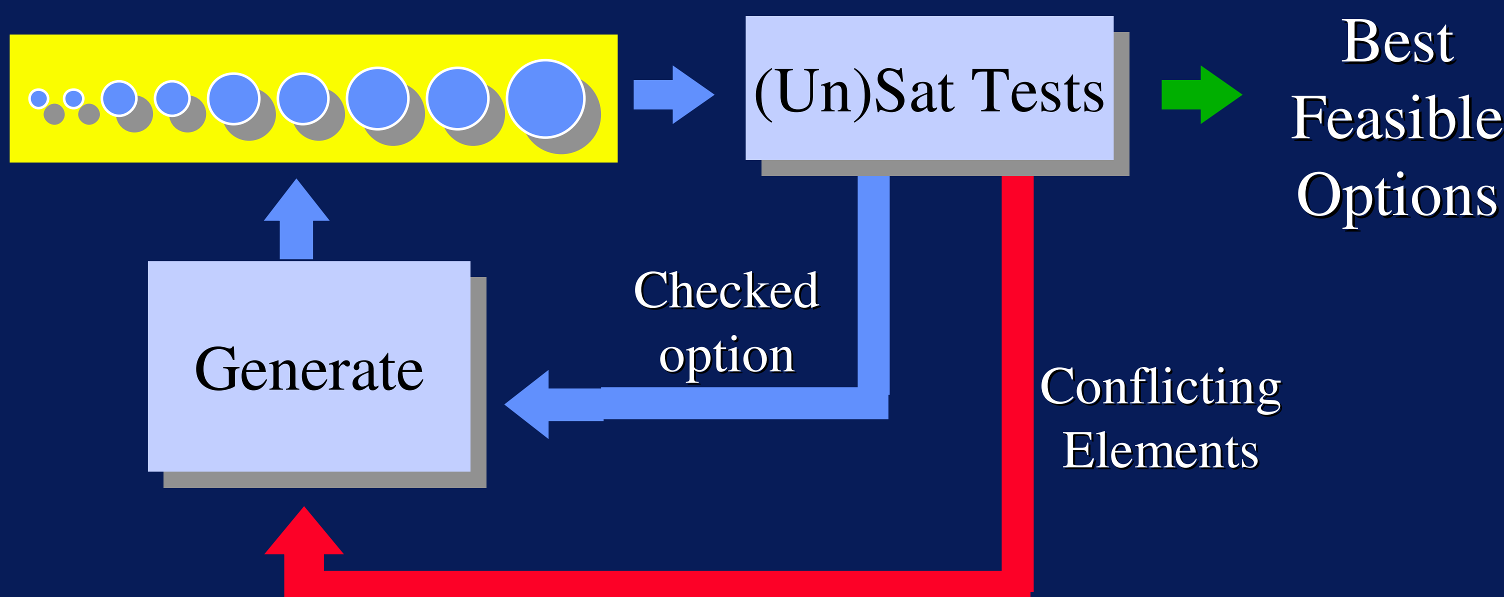
OPSAT

Generate Best Options:

-

Test Against Constraints:

-



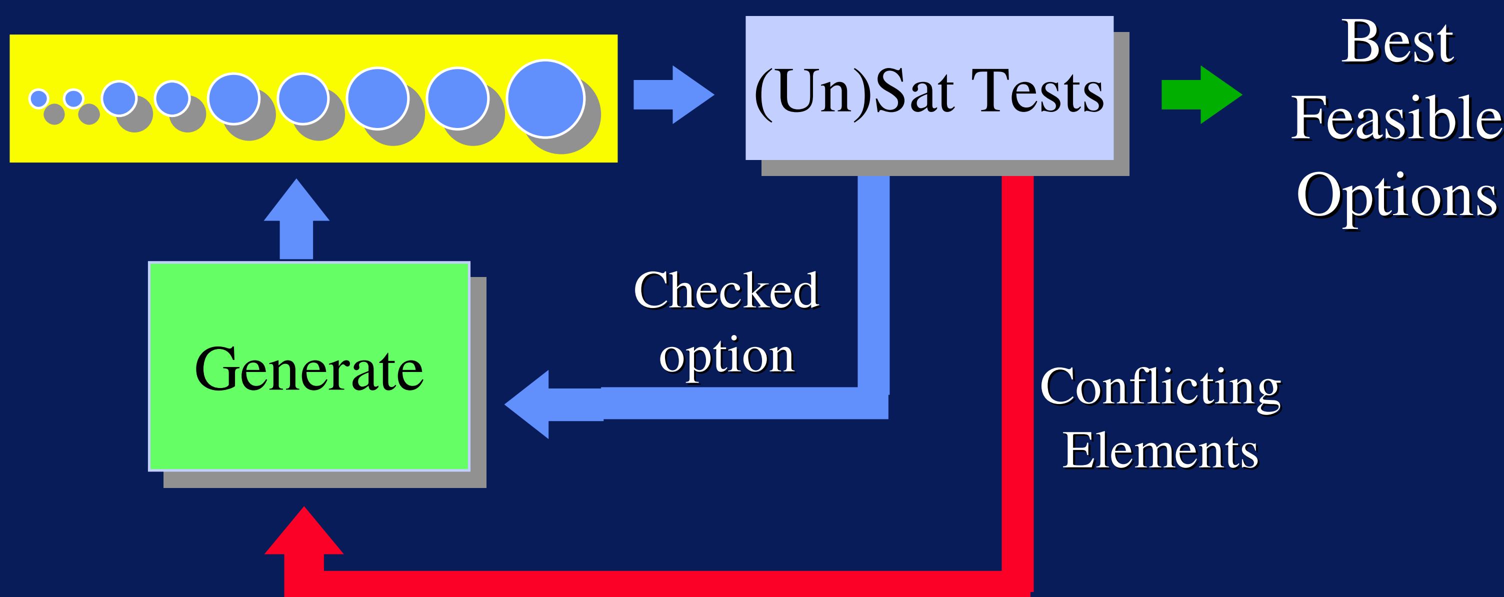
OPSAT

Generate Best Options:

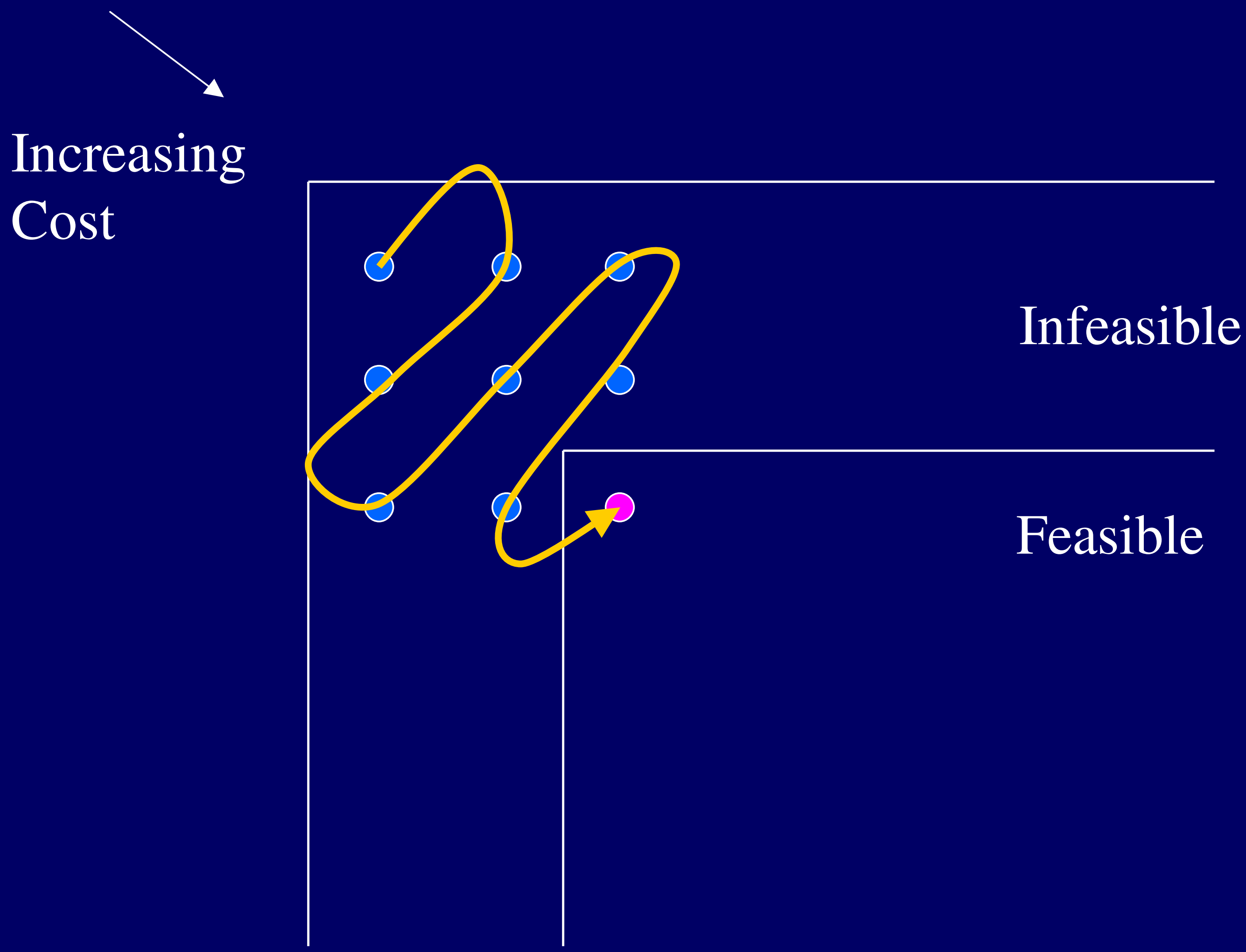
- Conflicts generalize test to leap over leading infeasible options

Test Against Constraints:

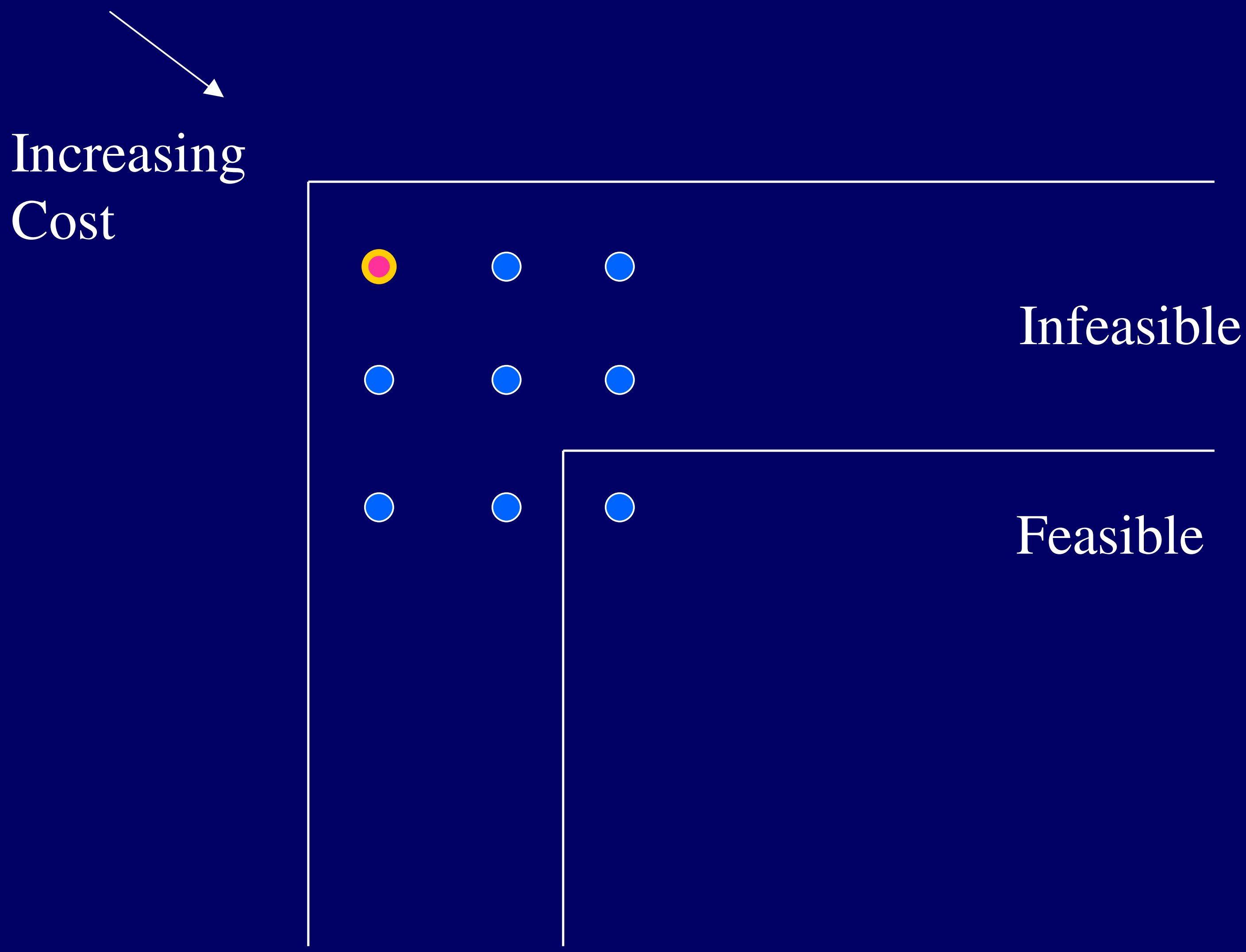
- Directed towards satisfying most constraints



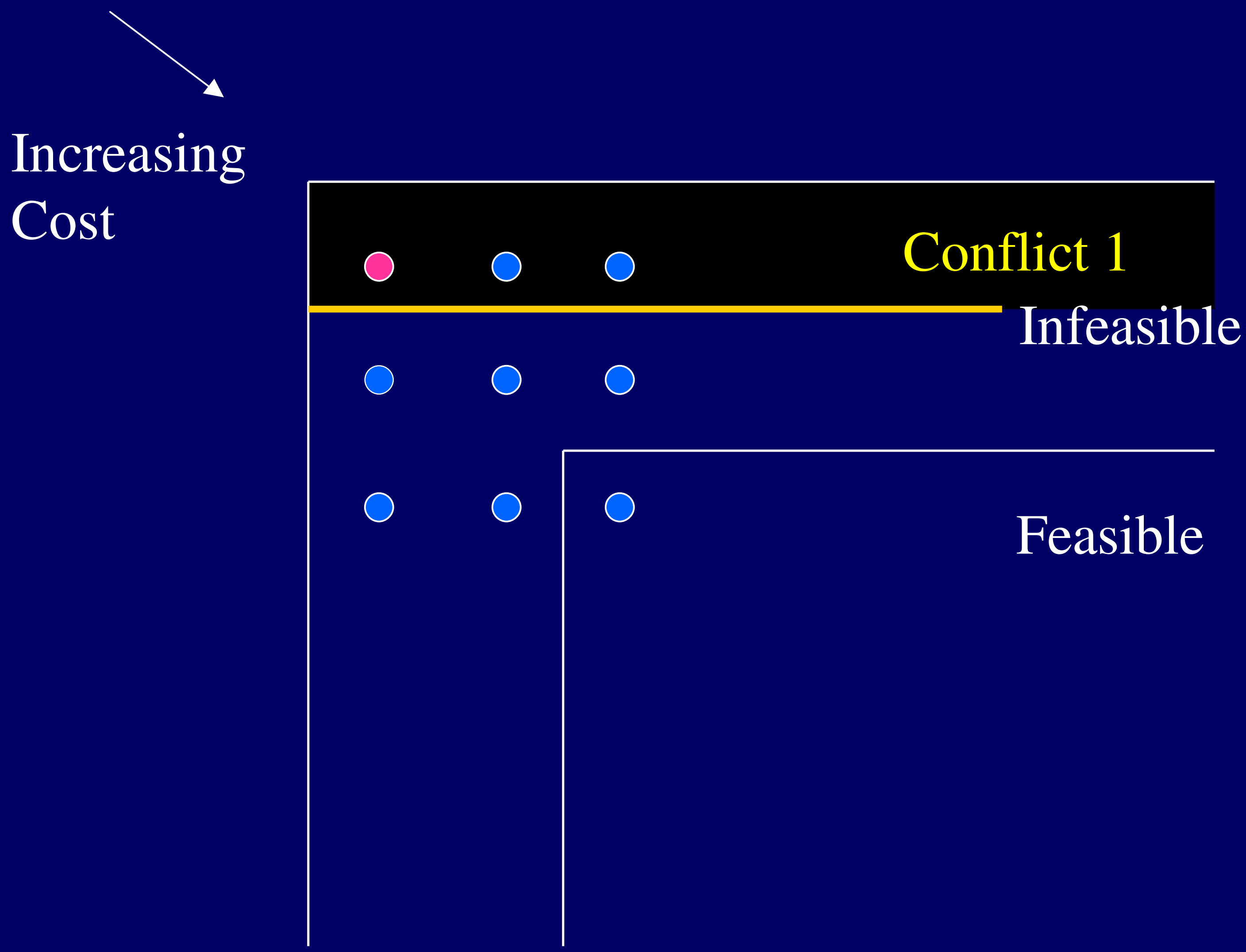
A*



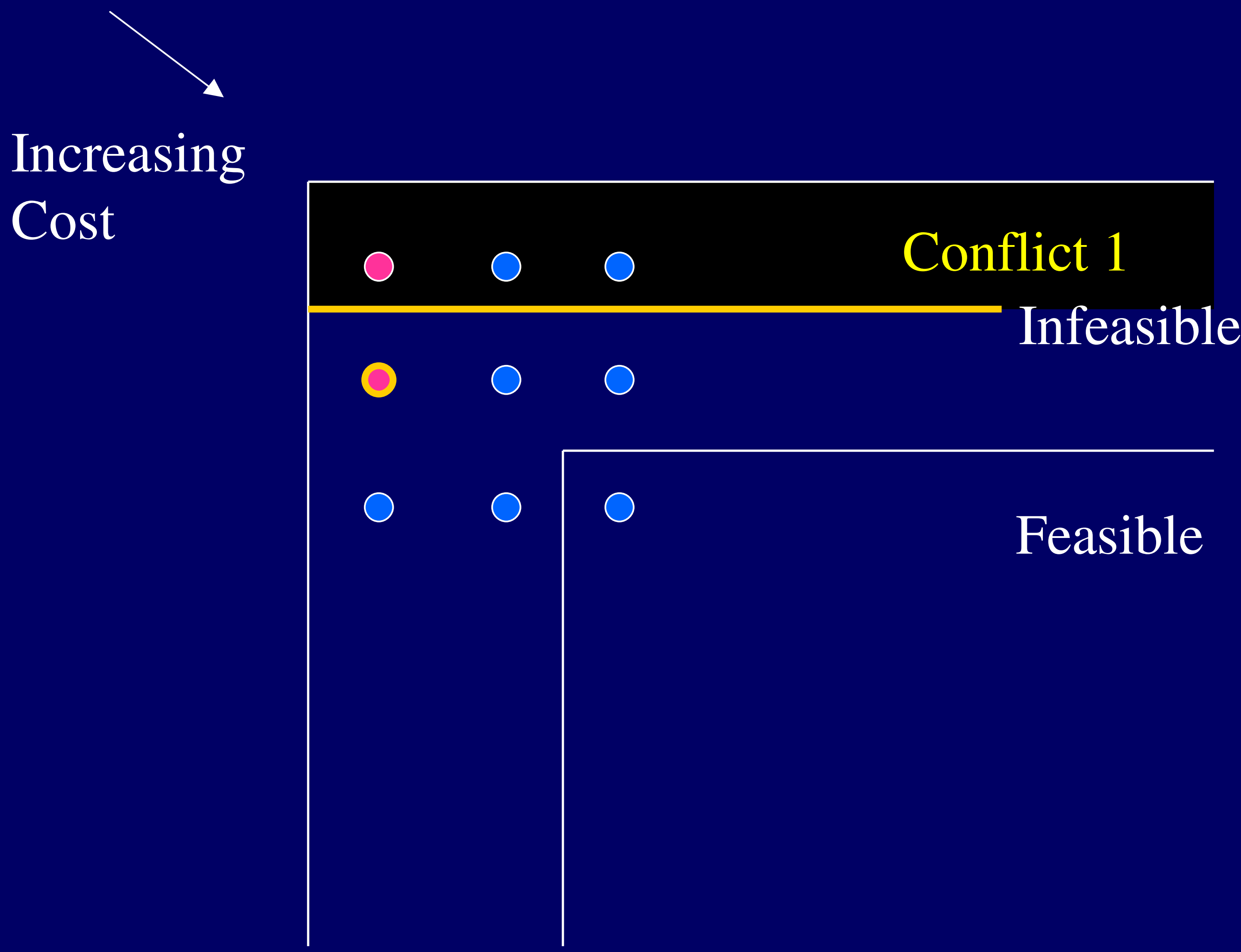
Conflict-directed A*



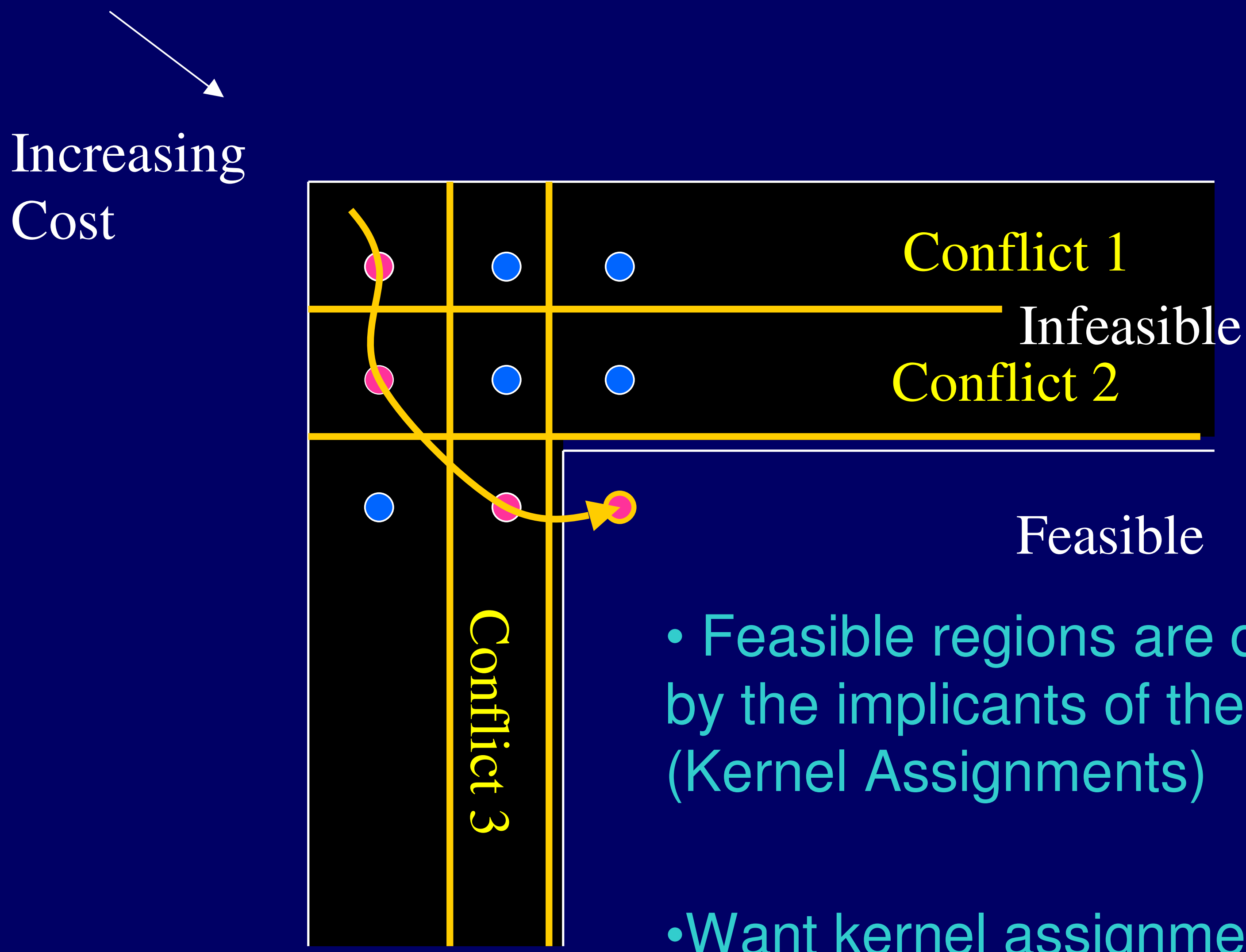
Conflict-directed A*



Conflict-directed A*



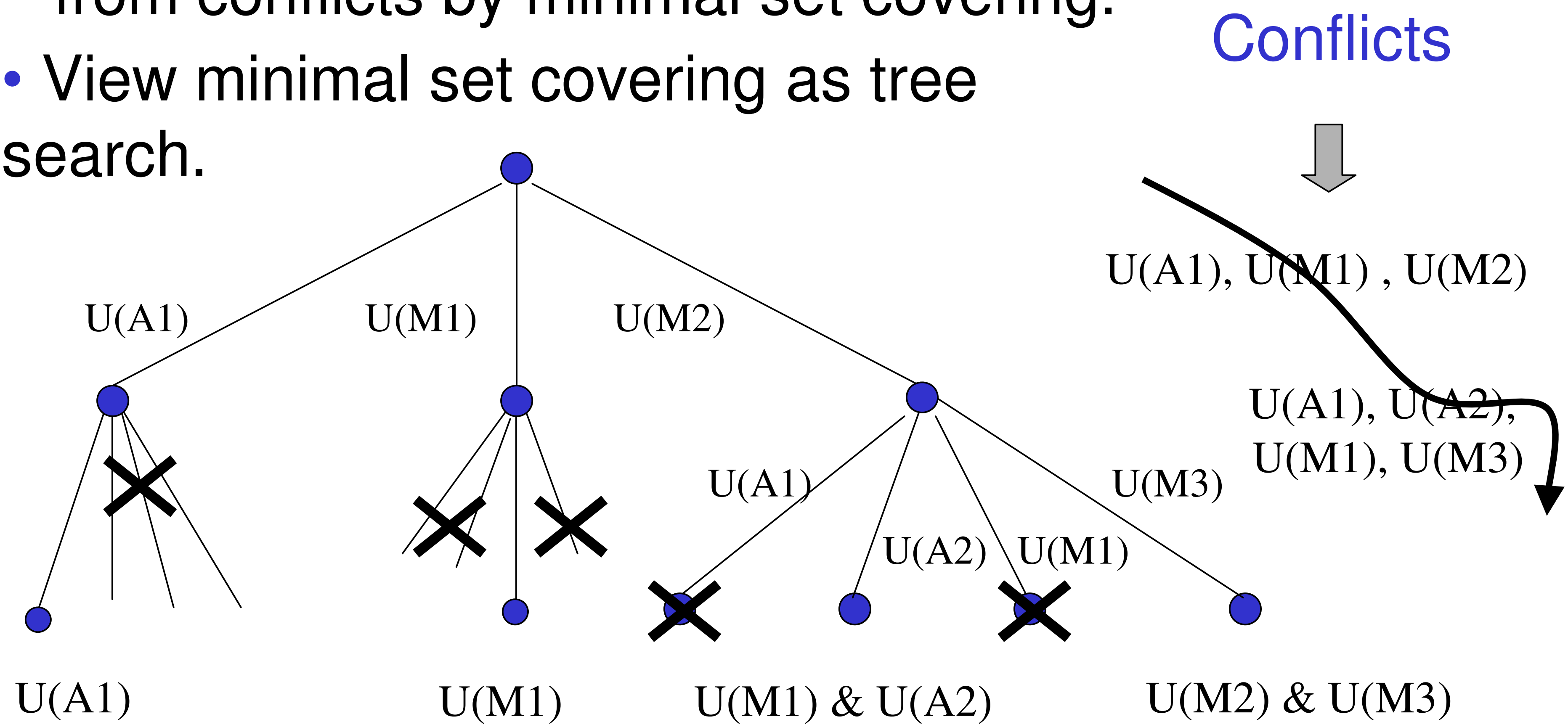
Conflict-directed A*



- Feasible regions are described by the implicants of the conflicts (Kernel Assignments)

- Want kernel assignment containing the best cost state.

- Kernel assignments are generated from conflicts by minimal set covering.
- View minimal set covering as tree search.

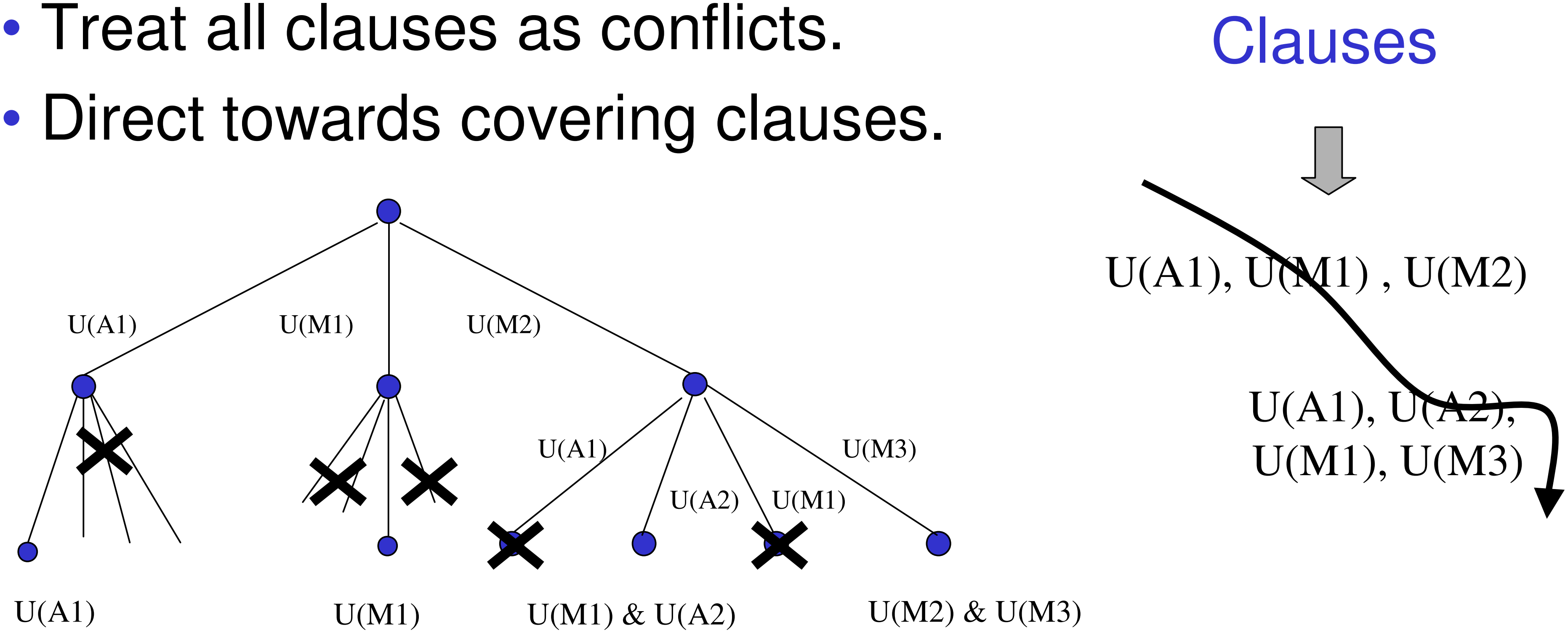


Conflict-directed A*:

- To find best kernel, expand tree in best first order, exploiting preferential independence, preserve systematicity
- Explore subspace of kernel in best first order.
- Test with Incremental Sat algorithm (DPLL + TMS)

How do we unify Generate and Test phases?

- Treat all clauses as conflicts.
- Direct towards covering clauses.



Clause-directed A*:

- Search in best first order, exploiting preferential independence.
- All else equal, direct towards assignments covering most clauses.
- Perform incremental unit propagation after each assignment.

➔ Produces best cost prime implicants.

Recent Publications: Optimal CSPs & OpSat

Using conflicts to optimally direct the selection of decision variables.

- Williams, B.C. and R. Ragno, "Conflict-directed A* and its Role in Model-based Embedded Systems," *to appear* Special Issue on Theory and Applications of Satisfiability Testing, Journal of Discrete Applied Math.

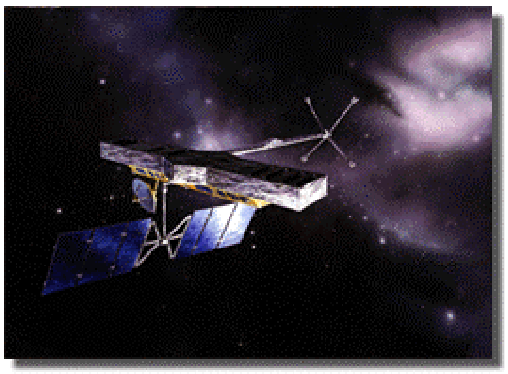
Unifying Generation and SAT Testing through Clause-direction

- Ragno, R. "Clause-directed A*," Master's Thesis, MIT EECS

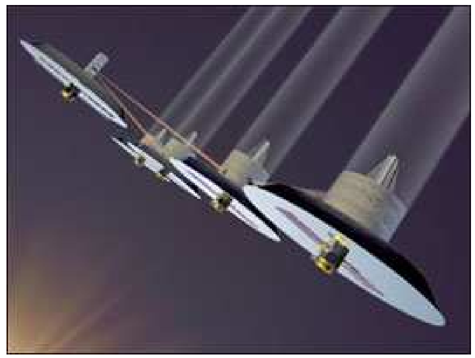
Demonstration 1: Interferometer Testbed

Objective: Successful ground test-bed demonstration:

1st step toward broader acceptance.

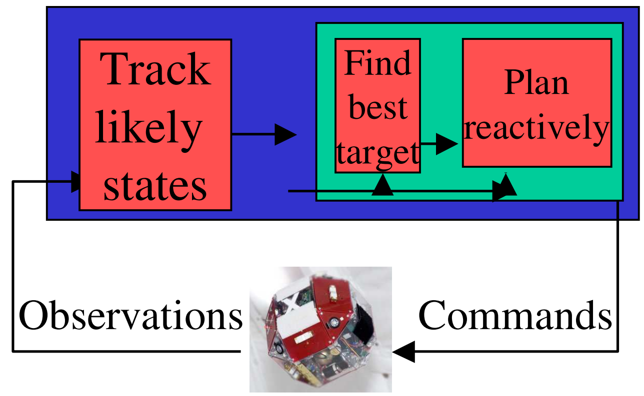


SIM



TPF

Deductive Controller
(lisp Livingstone)



Collaborators: JPL Caltech

Publication:

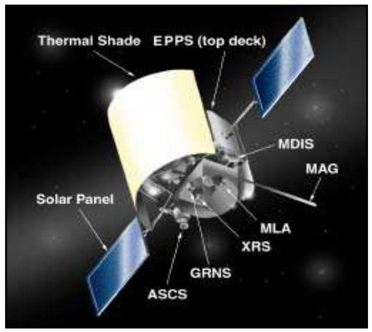
- Ingham, M., B. Williams, T. Lockhart, A. Oyake, M. Clark, A. Aljabri, "Autonomous Sequencing and Model-based Fault Protection for Space Interferometry," International Symposium on AI and Robotics in Space, June 2001.

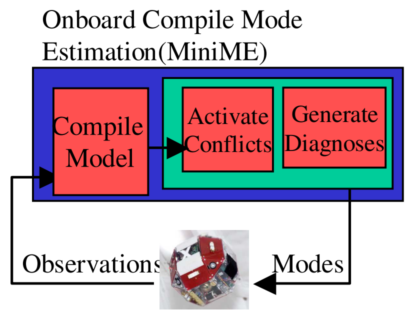
Terminated: Spring 01

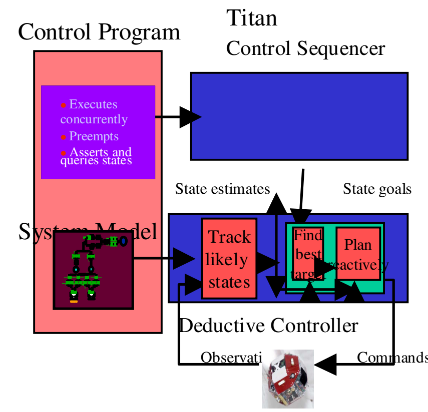
Demonstration 2 &3: Messenger On & Off Board

Objective: Demonstrate approach to Mode Estimation that is palatable to conservative missions.

Messenger Mission to Mercury







Collaborators: JHU APL Dave Watson, Mike Pekala…

Publication: Van Eepoel, J., B. Williams,S. Chung, "Improving Model-based Mode Estimation Through Offline Compiling," International Symposium on AI and Robotics in Space, June 2001.

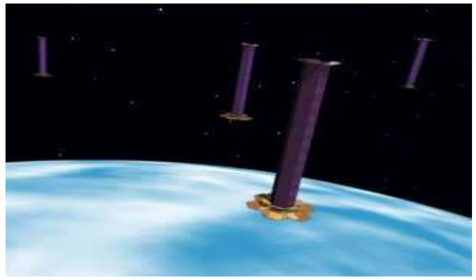
Status:

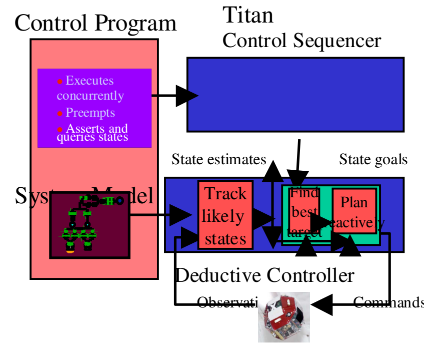
- Funding delayed to 8th month, FY 01,
- 1 month for Minime to reach Messenger PDR-> too late.
- Shifted to Titan ground station.
- Funding terminated after 4 months, still an excellent opportunity!

Demonstration 4: TechSat 21 onboard Model-based Execution

Objective: Model-based Programming on board, operate full mission

TechSat 21
Air force
Radar
Interferometer





Collaborators:

- MIT Space Systems Lab (Miller, Sedgwick, How, Fesq),
- MIT AI Lab (Shrobe, Ladagga, Sullivan, Roberston),
- JPL AIG (Chien, Rabideau, Sherwood), AFRL

Publication:

- Chien, S., R. Sherwood, M. Burl, R. Knight, G. Rabideau, B. Engelhardt, A. Davies, P. Zetocha, R. Wainright, P. Klupar, P. Cappelaere, D. Surka, B.C. Williams, R. Greeley, V. Baker and J. Doan, "The Techsat-21 Autonomous Sciencecraft Constellation," Int. Symp. on AI, Robotics and Automation in Space, St-Hubert, Canada, June 2001.

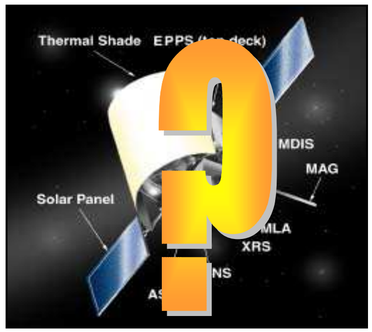
Status:

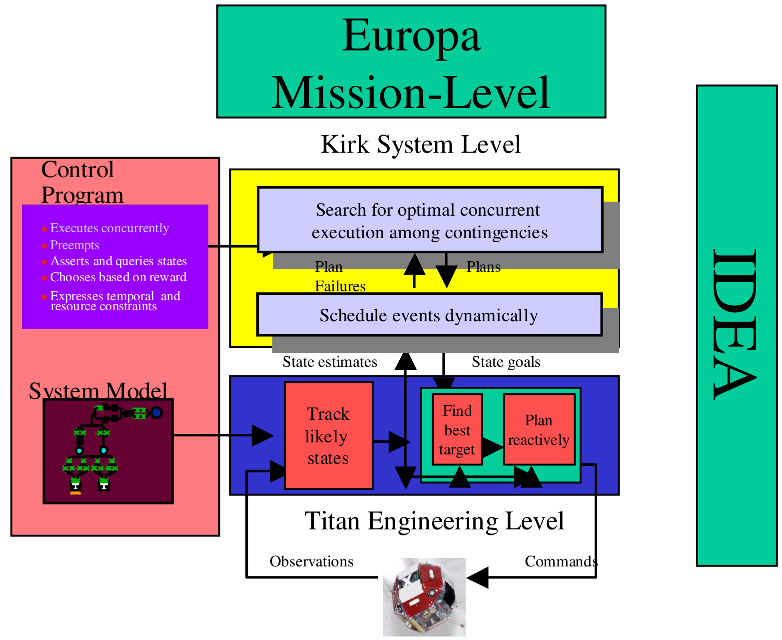
- Started FY 99 under AFRL funding, augmented by DARPA Mobies
- Replaced by L2, January, 2002.

Demonstration 5 & 6: ST –7 Concept Study
Model-based Programming at
Engineering & System Levels

Objective: Demonstrate Model-based Programming on board at Engineering and Systems Levels, operate full mission

NASA ST7
Mission
Phase A





Collaborators:

- JPL (Beam Group) AIG (Barret)
- JHU APL (Watson, Pekala)
- NASA Ames (Muscettola, Morris)

Publication:

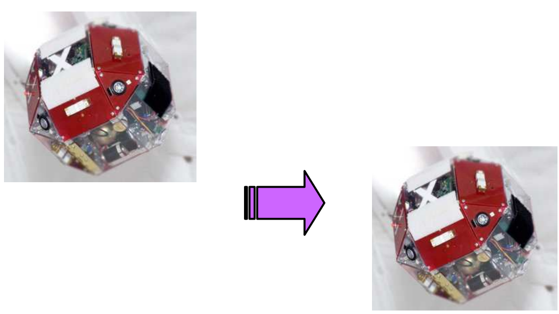
- Fesq, L., et al. “Model-based Programming for Robotic Spacecraft” *to appear*, World Space Congress, 2002.

Status:

- Completed integrated demo of System level (Kirk) and Eng. Level (Titan)
- Awaiting Mission level autonomy and IDEA for full integrated demo.

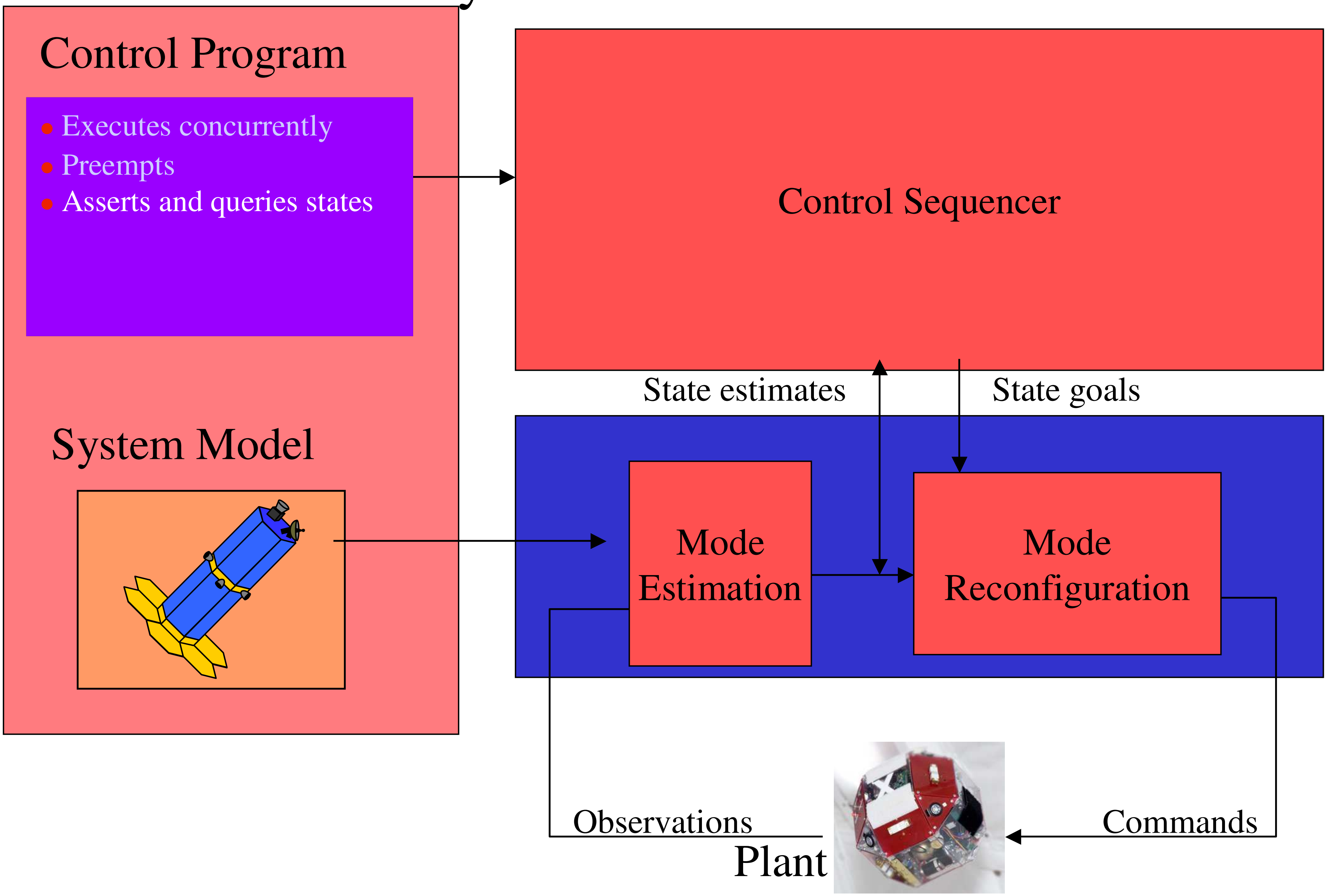
Directions: Spheres on ISS

- Working to demonstrate Titan in flight on docking maneuvers for MIT Sphere's Spacecraft within Intl. Space Station.
- Titan must manage mission in light of failures.

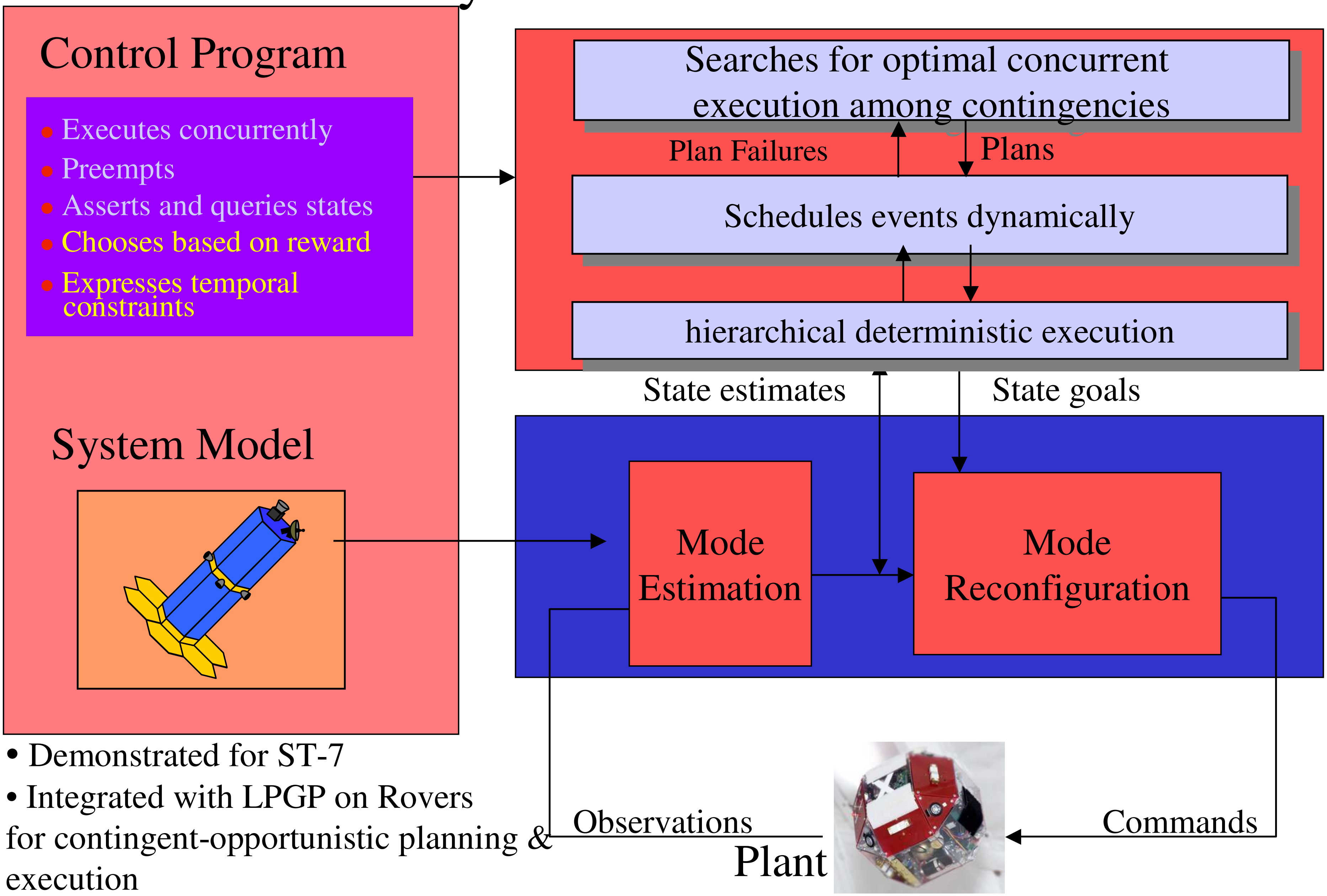


Funded by DARPA Orbital Express

Directions: Expanding Model-based Programming to the System-level: Titan + Kirk



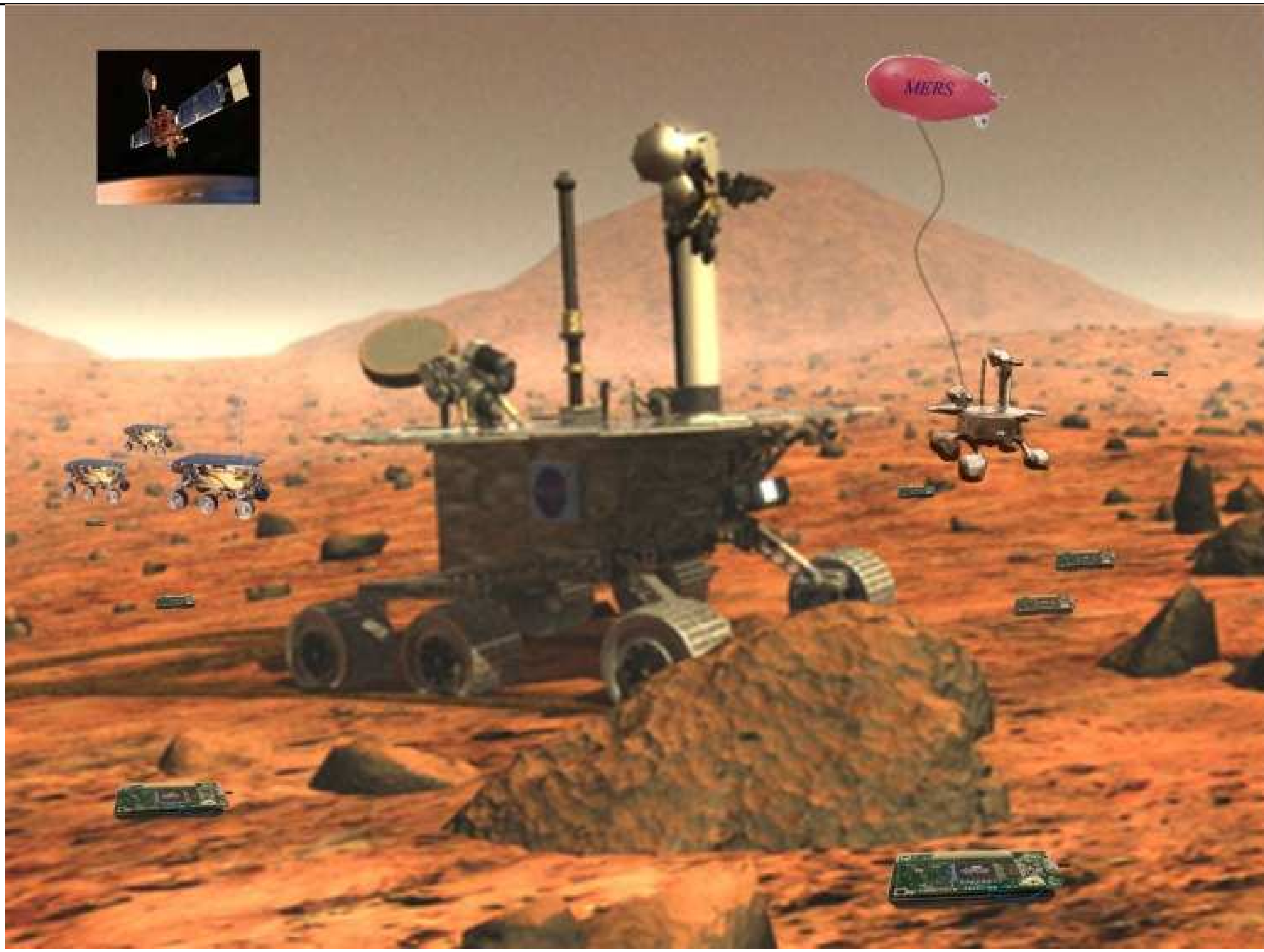
Directions: Expanding Model-based Programming to the System-level: Titan + Kirk



• Demonstrated for ST-7

• Integrated with LPGP on Rovers for contingent-opportunistic planning & execution

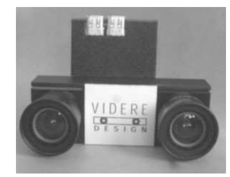
Heterogeneous Cooperative Robotics



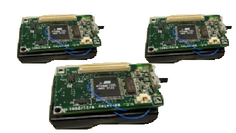
- Orbiter
- Tethered Blimp
- Mobile Lander
- Scout Rovers
- Sensor Network

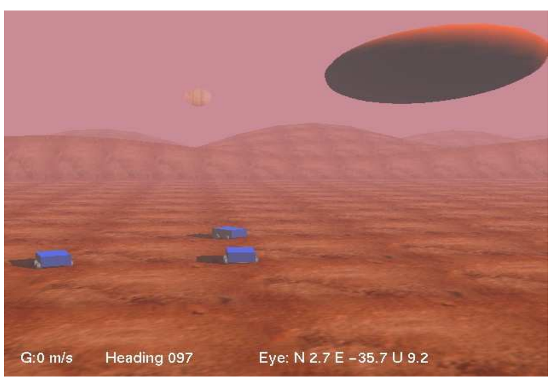
Indoor Testbed

- One ceiling mounted stereo camera “the blimp”
- 3 ATRV jr.
- 1 ATRV
- Rover Sensors
  - Stereo camera head
  - Sonar array
  - Laser range scanner
  - DGPS
  - Wheel encoders
  - Digital compass



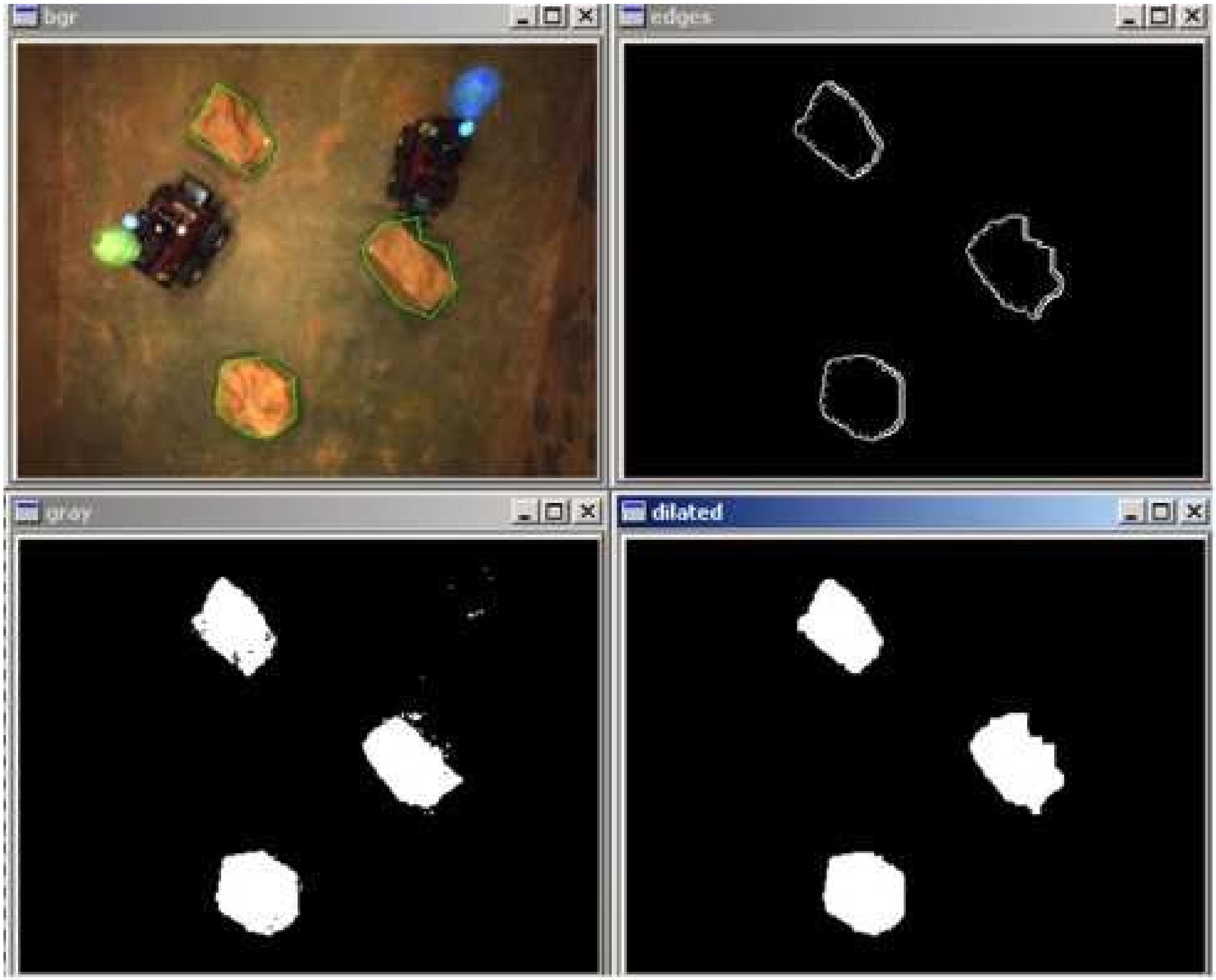
- Motes Sensor Networks



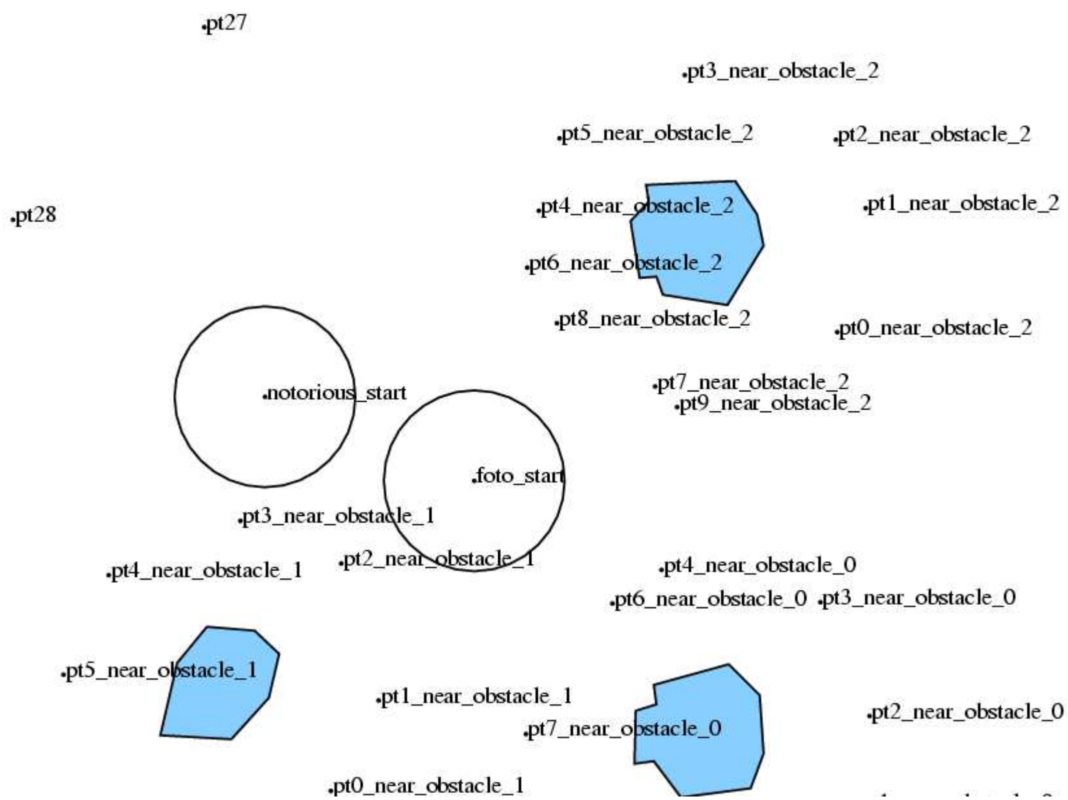


Rover Sim

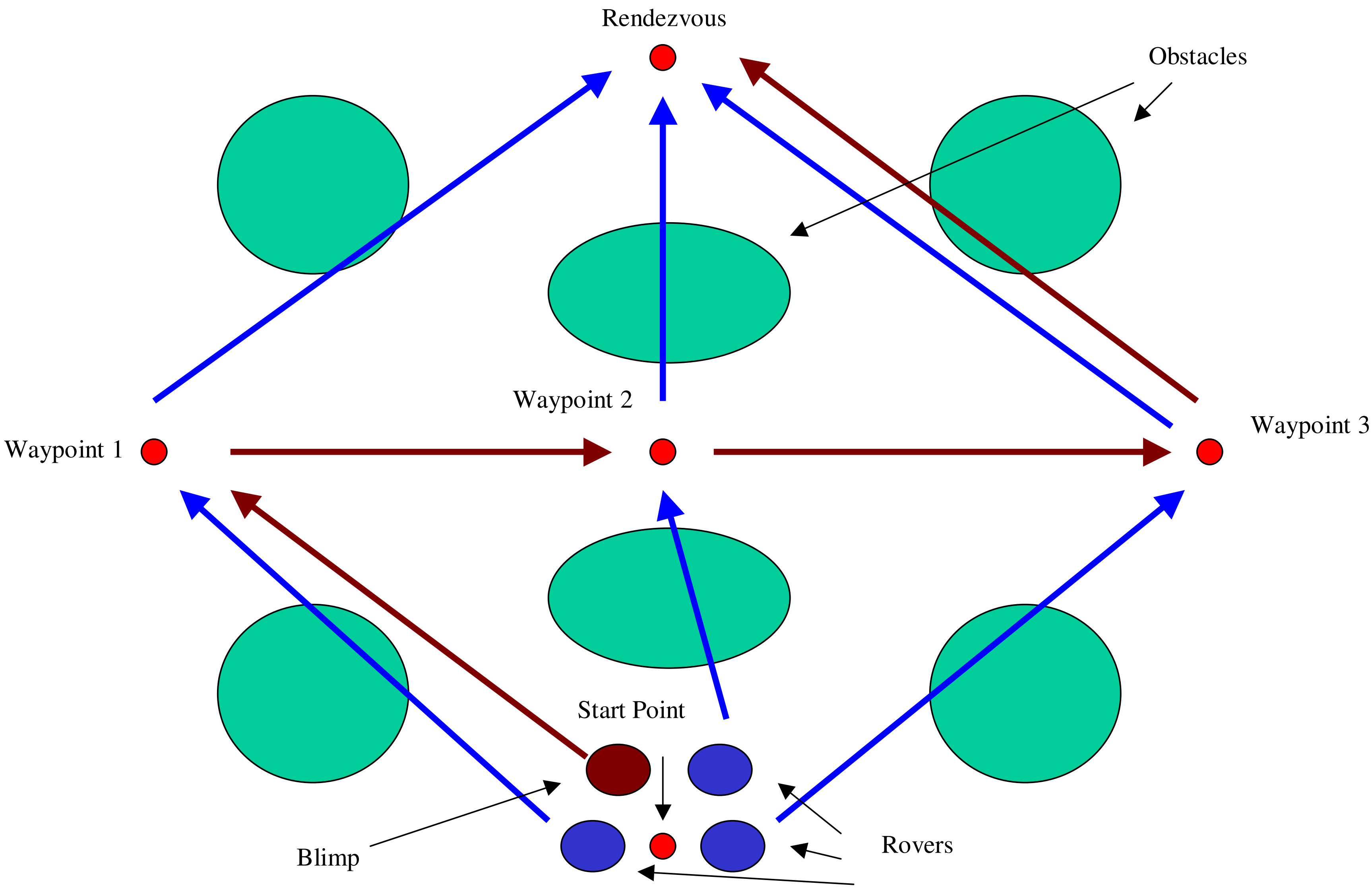
Obstacle Detection



Map Generation



RMPL Generated Mission



demo on sim

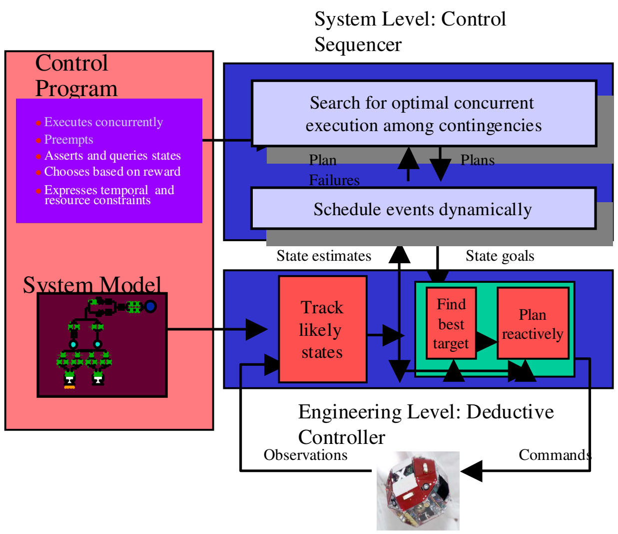
Rover tracking during execution

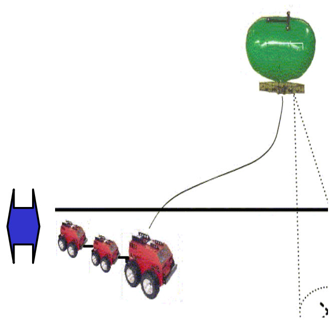
Issues

As a university, how can we effectively establish a path for our technology to MSL, and other missions, and how do we secure funds to support this?

MIT

Model-based Execution for Space Vehicles





MIT Cooperative Rover Testbed

TASK OBJECTIVES:

Develop model-based embedded programming languages that think from commonsense models in order to robustly command, monitor, diagnose and repair collections of robotic explorers.

TECHNICAL INNOVATIONS:

•RMPL language reads and writes hidden state variables as if directly observable and controllable

•Titan executive "reads" state by automatically deducing it from sensor information.

• Titan executive "sets" state by planning command sequences that move to the specified state.

SPONSOR: NASA Code-R (CETDP)

DEVELOPMENT TEAM: MIT, [APL IS]

Milestones	FY02	FY03	FY04
Develop RMPL Compiler	X		
RMPL executive	X	X	
Distributed RMPL executive		X	X
•Demonstrate on Distributed Space System Testbed(s)		X	X

NASA RELEVANCE:

•Provides high assurance software for space missions.

• Offers robust capabilities for command execution and fault management.

•Speeds time for development and testing of flight software.

•Dramatically expands ability to robustly respond to novel situations.

Model-based Programming as Estimating, Planning and Executing based on Hidden State

Brian C. Williams

Artificial Intelligence and Space Systems Labs

Massachusetts Institute of Technology

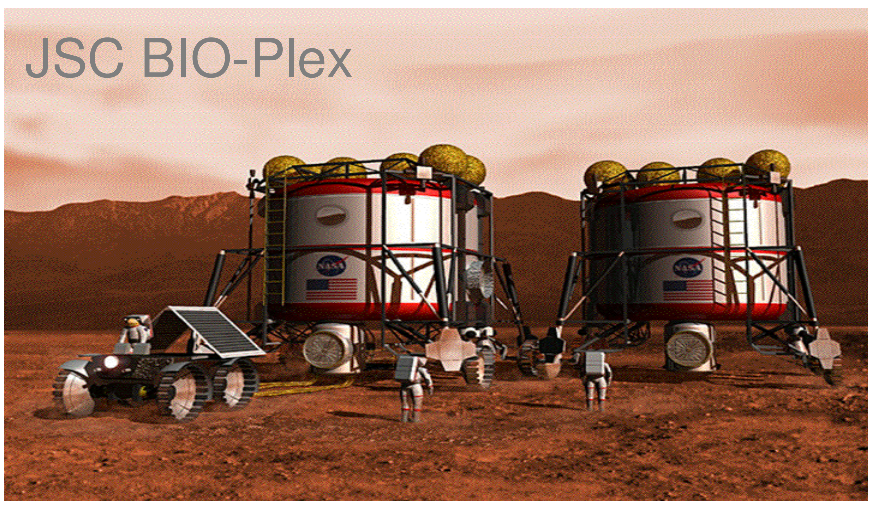
IS Program Review

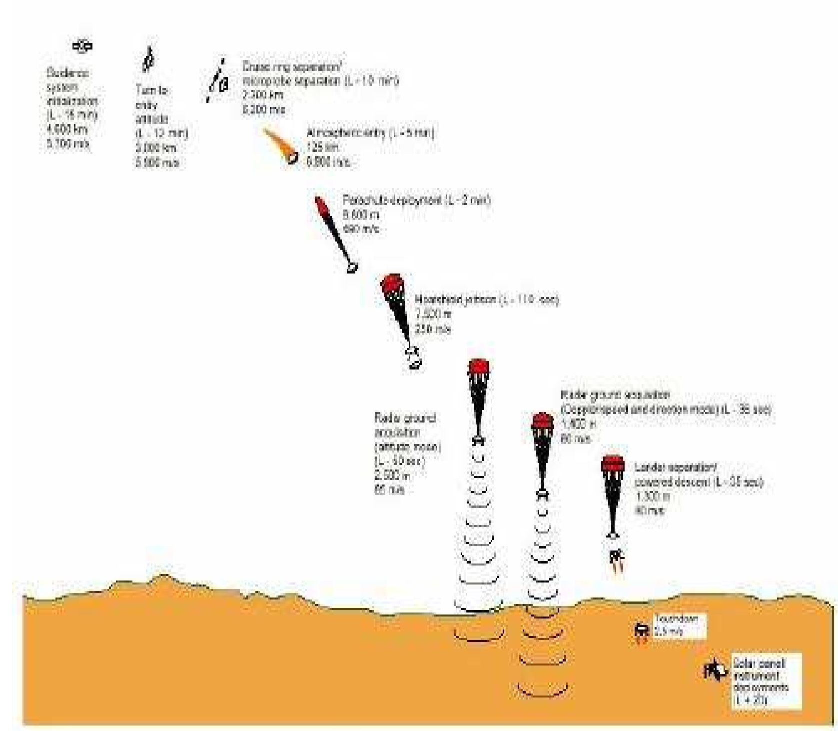
September 5th, 2002

Objective

Create a hybrid estimation, monitoring, diagnosis and model learning capability for physical devices that exhibit complex discrete and continuous behaviors.

DEMONSTRATION:





Entry, descent and landing

Mars Entry, descent & Landing

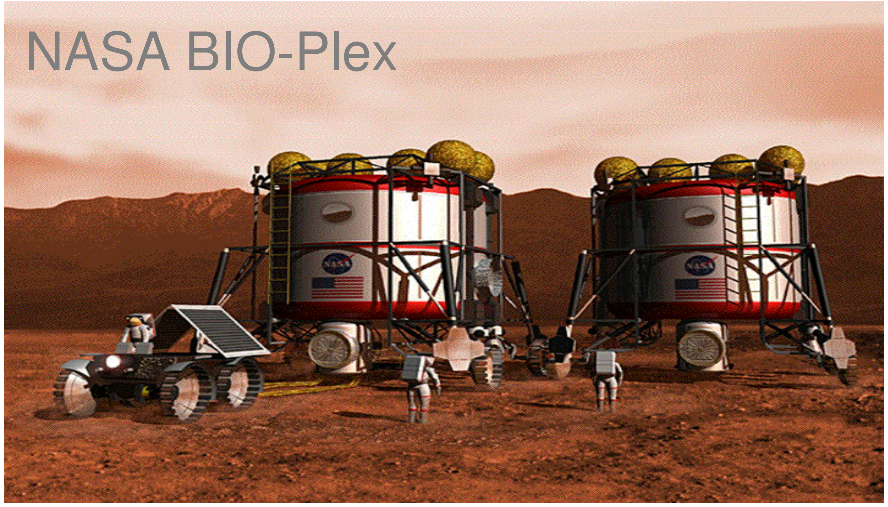
A Hybrid Discrete/Continuous System for Health Management

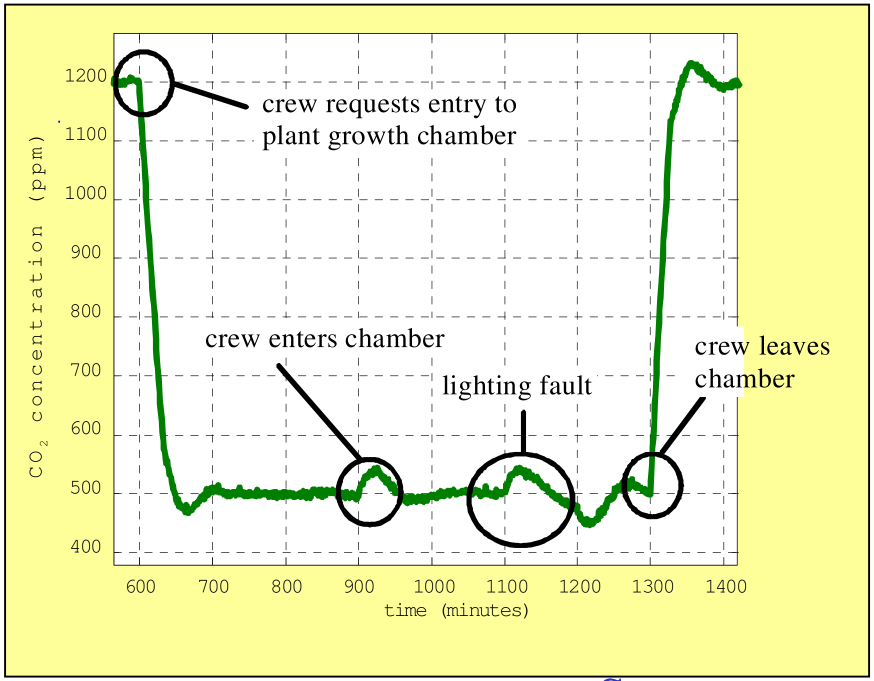
- Failures can manifest themselves through a coupling of a system's continuous dynamics and its evolution through different behavior modes

  ⇒ *must track over continuous state changes and discrete mode changes*

- Symptoms are initially subtle; on the same scale as sensor/actuator noise

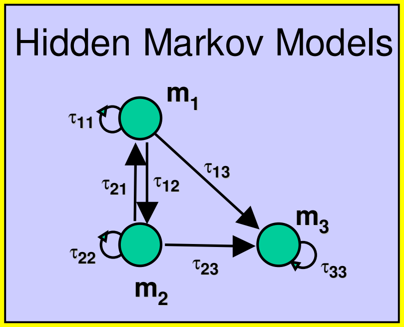
  ⇒ *need to extract mode estimates from subtle symptoms*





Support:
- NASA IS

Hybrid Plant Model for HME



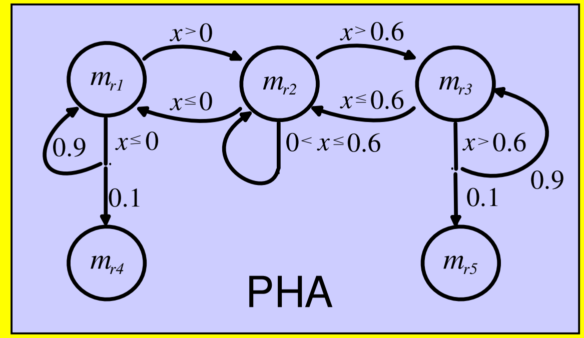
+

Continuous Dynamics

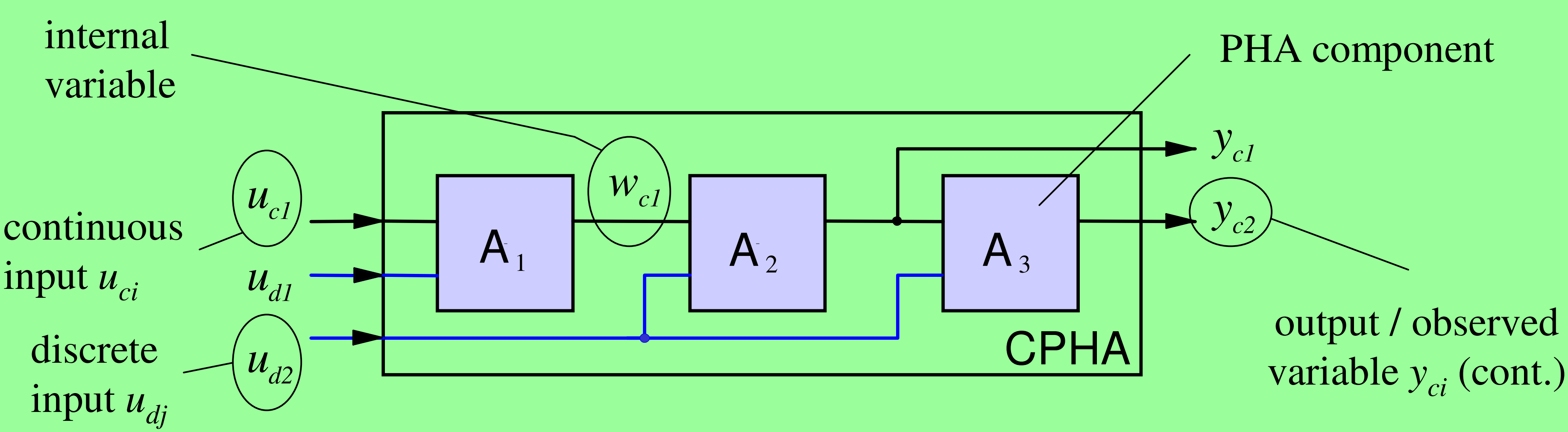
$$m_1 : \begin{cases} x_c(k+1) = f_{c1}(x_c(k), u_c(k), v_c(k)) \\ y_c(k) = g_{c1}(x_c(k), v_c(k)) \end{cases}$$

$$\vdots$$

$$m_i : \begin{cases} x_c(k+1) = f_{ci}(x_c(k), u_c(k), v_c(k)) \\ y_c(k) = g_{ci}(x_c(k), v_c(k)) \end{cases}$$

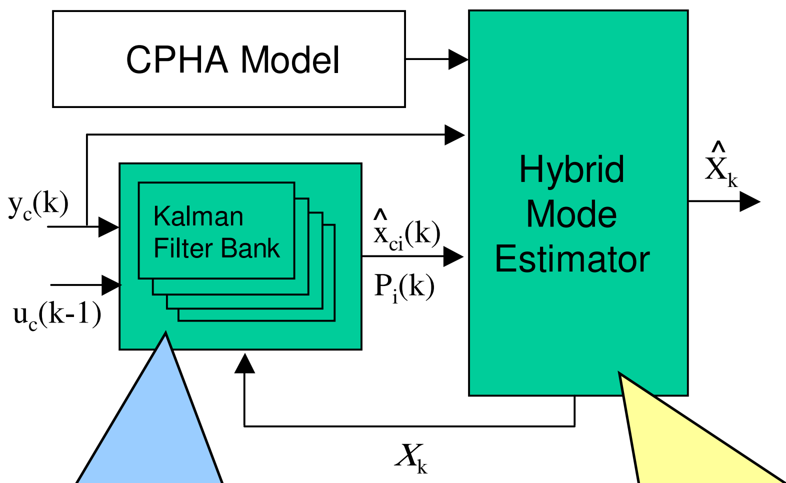


Concurrent Probabilistic Hybrid Automaton (CPHA):



Hybrid Mode / State Estimation

sensor signals $\mathbf{y}_c$ and control inputs $\mathbf{u}_c$, $\mathbf{u}_d$



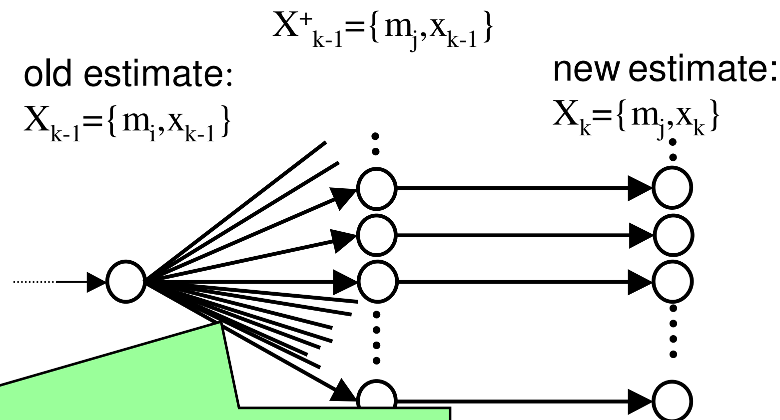
estimated mode/state $\mathbf{x} = \{\mathbf{x}_d, \mathbf{x}_c\}$ and its belief state $h[\mathbf{x}]$

Hybrid State Estimator
Maintains the set of most likely hybrid state estimates as a set of trajectories.

Hybrid Mode Estimator:
Determines for each trajectory the possible transitions, and specifies (dynamically) the candidate trajectories to be tracked by the continuous state estimators.

Hybrid Mode / State Estimation

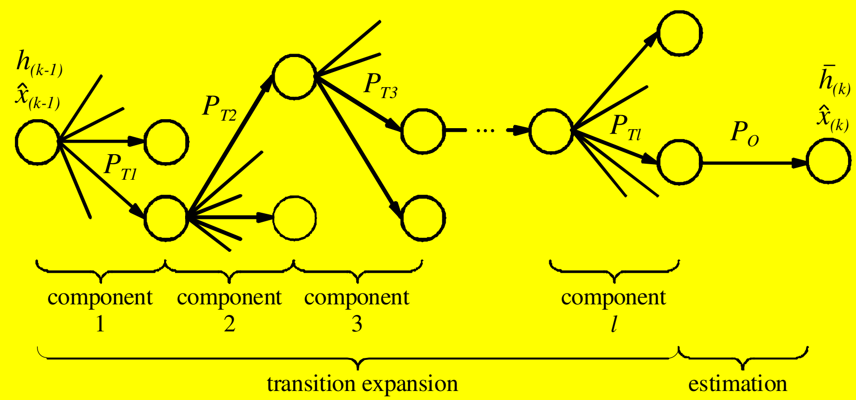
1. *HMM-style belief state update determines the likelihood for each discrete mode transition.*
2. *Kalman-filter-style update determines likelihood of continuous state evolution.*



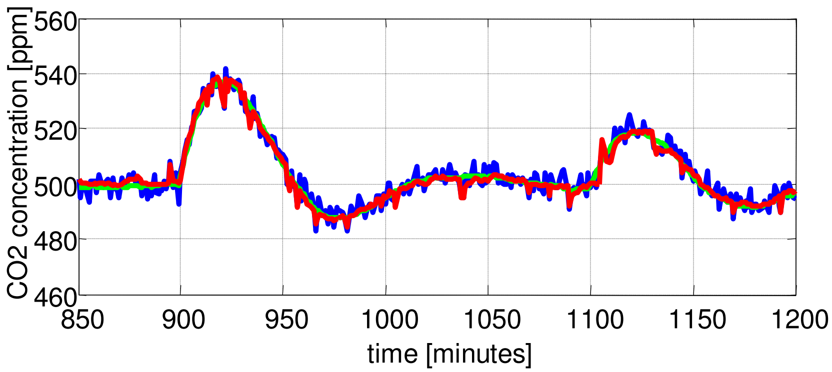
transitions at each time step is very large:
e.g. model with 10 components, each with 3 successor modes has $3^{10} = 59049$ possible successor modes *for each trajectory*!

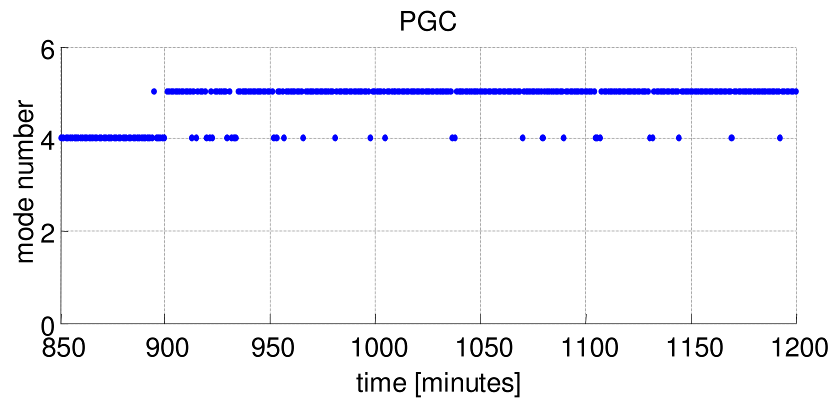
How to handle the exponential blowup?

- Generalize beam search to track the most promising hybrid states.
- Factor state space into lower dimensional subspaces through automated decomposition and filter synthesis.



Simulation Result





components**:** 6

(FR1, FR2, PIV1, PIV2, LS, PGC)

total # of modes: 9600

fringe size: **20** (400 estimation steps):

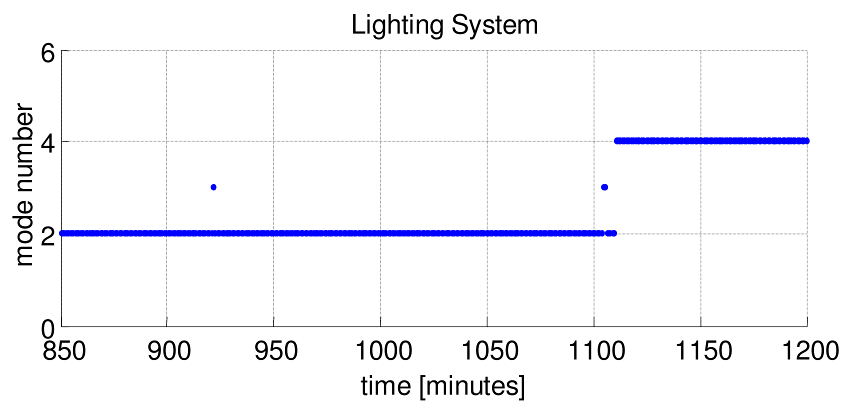
average candidates: 90.2 (< 1% !)

max. candidates: 428 (< 5 %!)

filter calculations: 242

filter executions: 36050

average runtime: ~1 s/step (PII-400, 128mb)



Recent Publications

Hybrid Mode Estimation:

- Hofbaur, M. W. and B.C. Williams, "Mode Estimation of Probabilistic Hybrid Systems," International Conference on Hybrid Systems: Computation and Control, March, 2002.

Hybrid Expectation Maximization (preliminary):

- Melvin Henry, *Simulators that Learn: Automated Estimation of Hybrid Automata*, June 2002

Hybrid Decomposition (preliminary):

- Hofbaur, M. W. and B. C. Williams, "Hybrid Diagnosis with Unknown Behavioral Modes," International Workshop on Principles of Diagnosis, Austria, May 3-5 2002.

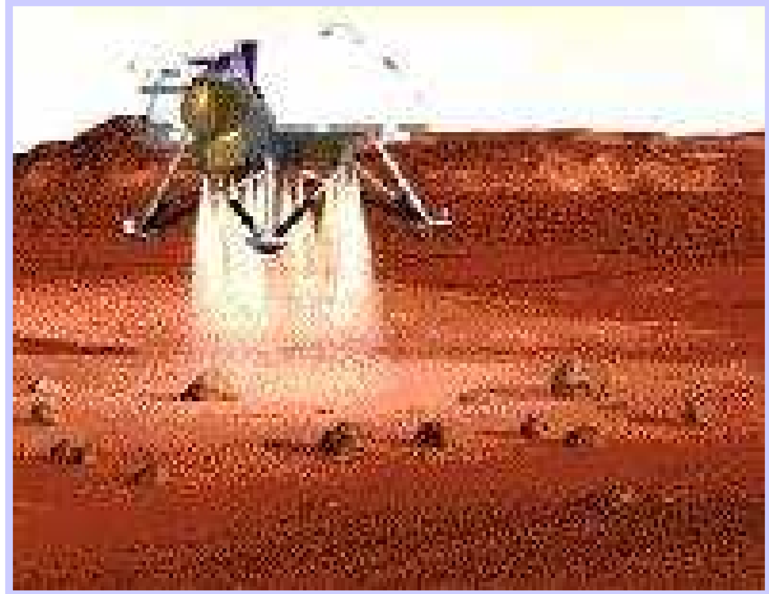
Future Directions

- Model-Learning as Hybrid EM
- Automated Decomposition of HPCA using Dissents
- Model-based Hybrid Execution

To Address the Scope of Mars 98

Polar Lander Leading Diagnosis:

- Legs deployed during descent.
- Noise spike on leg sensors latched by software monitors.
- Laser altimeter registers 50ft.
- Begins polling leg monitors to determine touch down.
- Latched noise spike read as touchdown.
- Engine shutdown at ~50ft.



Responding to the failures of Mars Polar Lander and Mars Climate Orbiter is a Hybrid control problem.

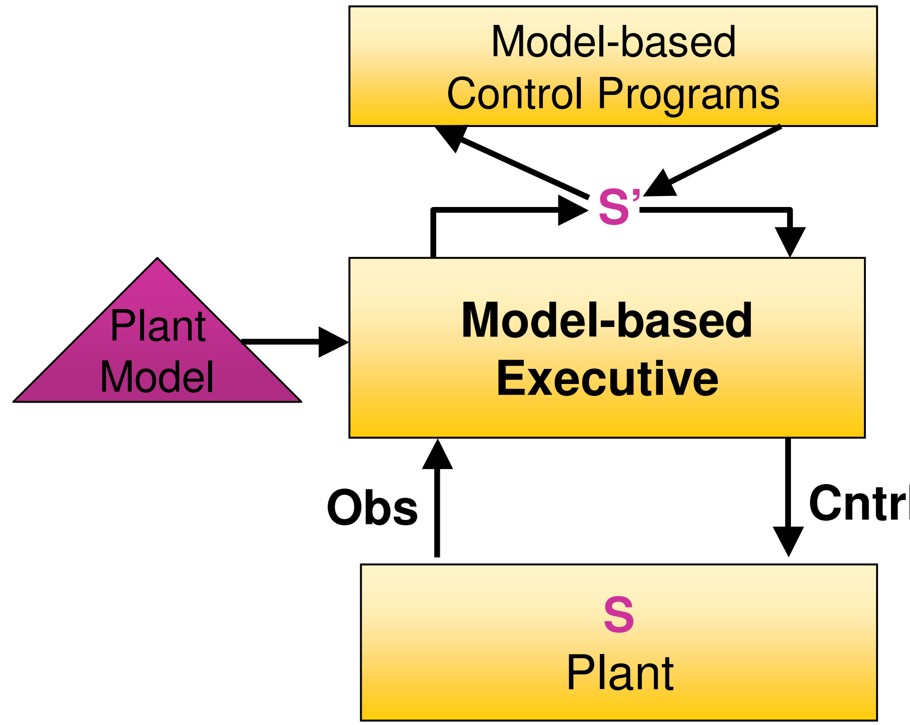
Idea: Support programmers with embedded languages that avoid commonsense mistakes, by reasoning from hardware models.

Reactive Model-based Programming

Hybrid Model-based Programming

Hybrid Model-based Programs:

- Extend to include assertions and queries on continuous states.

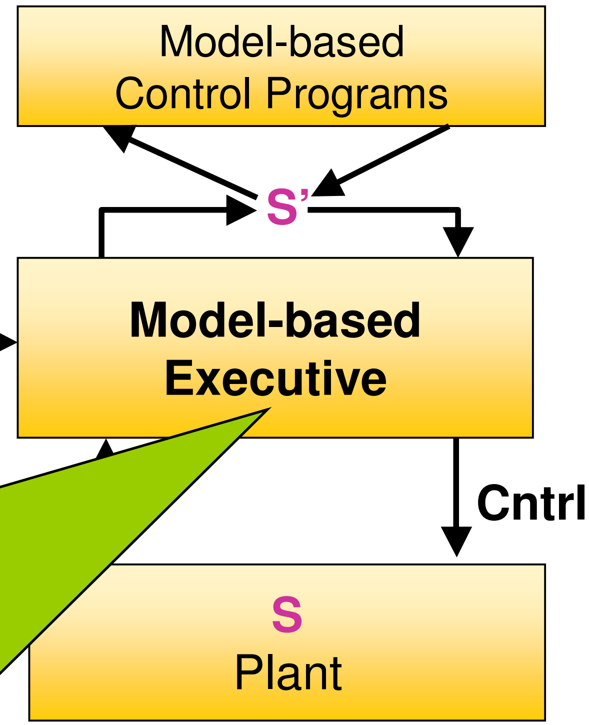


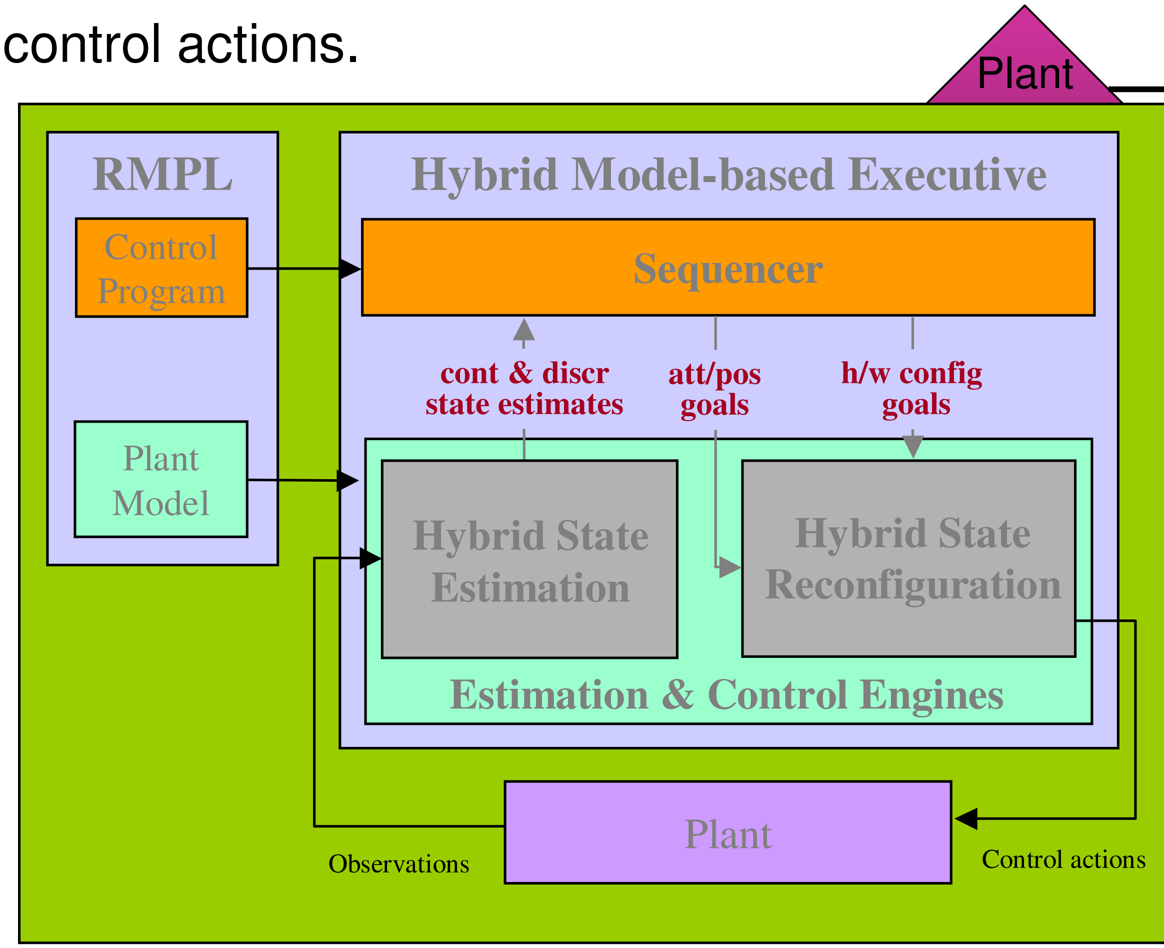
Support:
- NASA IS

Hybrid Model-based Programming

Hybrid Executives:

- Deduce continuous as well as discrete states.
- Issue continuous as well as discrete control actions.



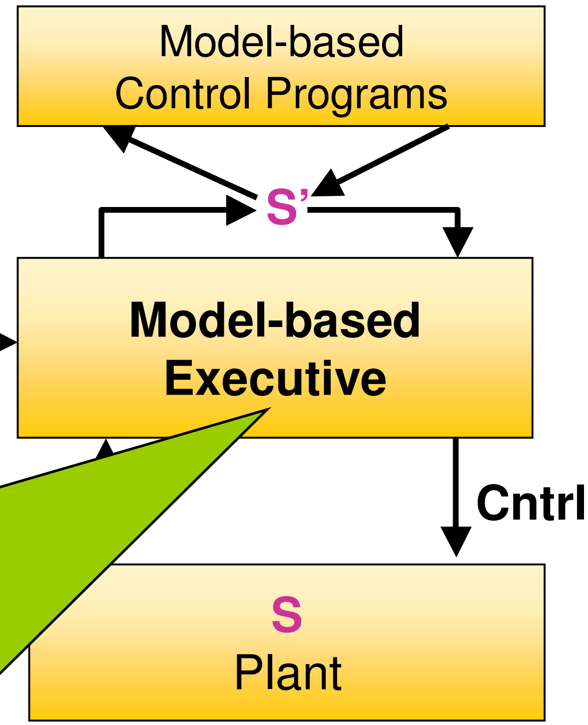


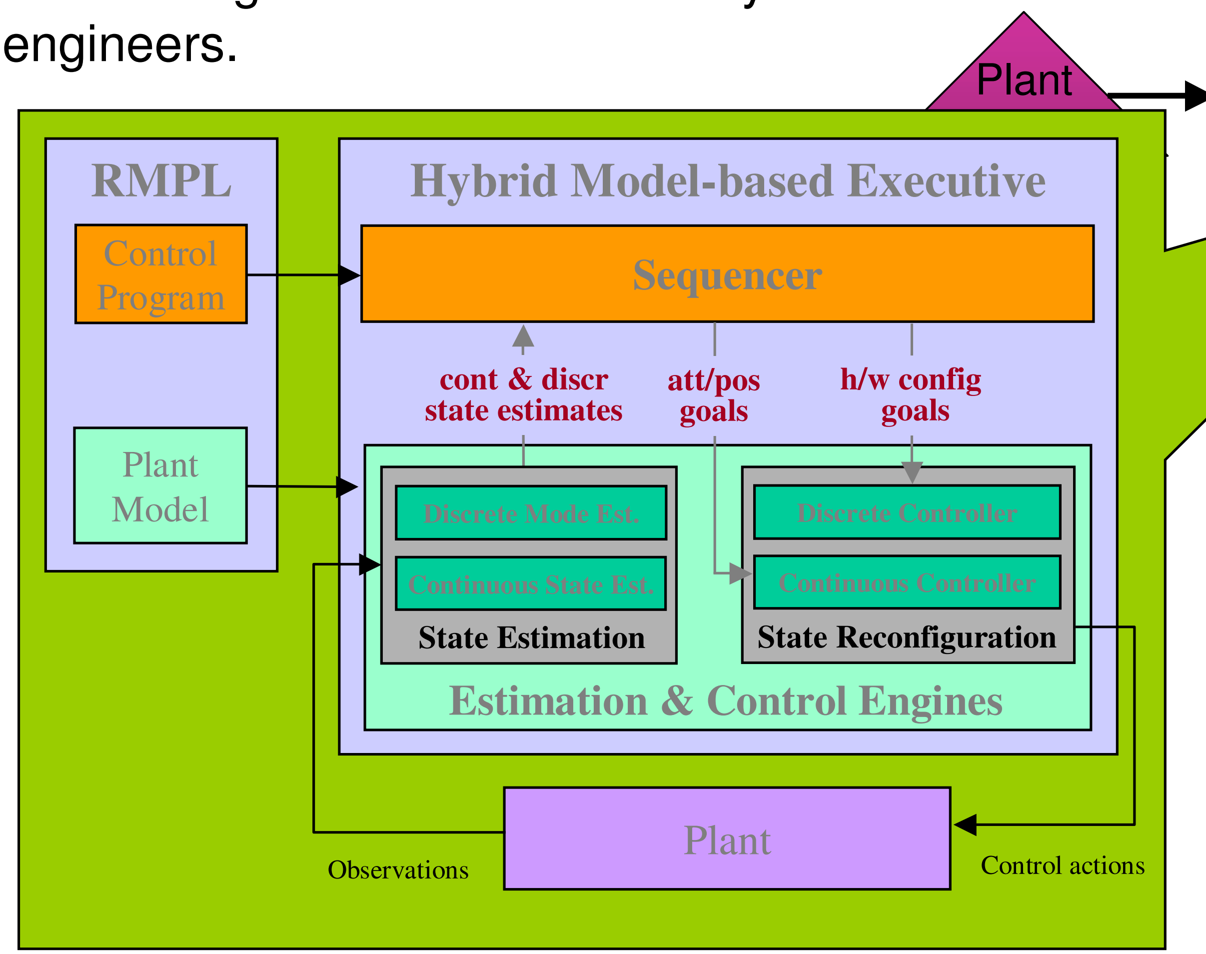
Support:
- NASA IS

Hybrid Model-based Programming

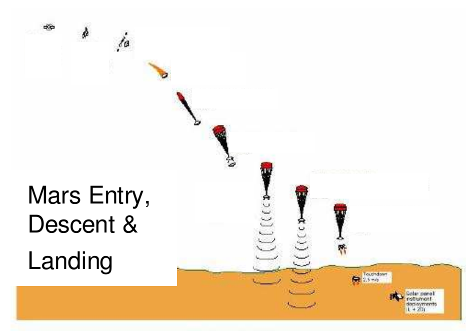
Hybrid Executives:

- Can hook into existing estimation and control approaches.
- Should target “comfort zone” of systems engineers.



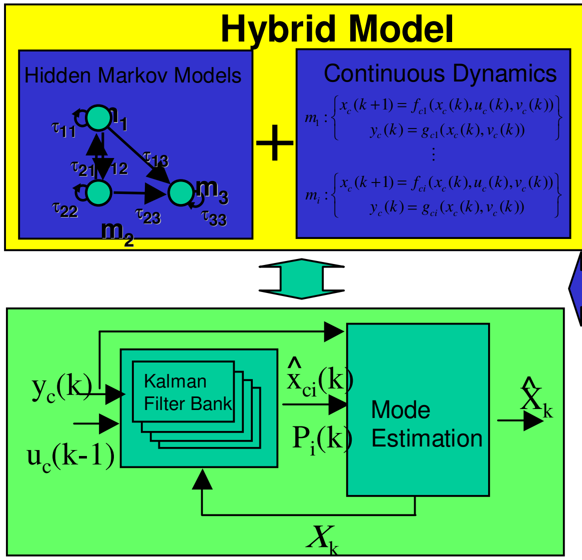


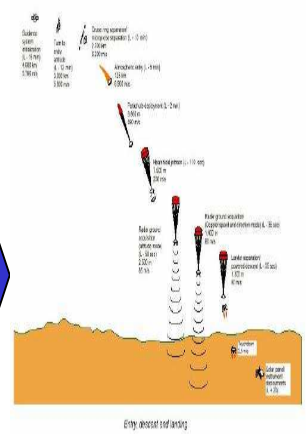
Demonstration:



MIT

A Hybrid Discrete/Continuous System for Health Management





Mars Entry, Descent & Landing

TASK OBJECTIVES:

Create a hybrid monitoring, diagnosis and model learning capability for physical devices that exhibit complex discrete and continuous behaviors.

TECHNICAL INNOVATIONS:

Dynamics are modeled as hybrid probabilistic concurrent automata (HPCA).

Monitoring, diagnosis, state tracking and model learning framed as elements of an Expectation Maximization algorithm for HPCA.

SPONSOR: NASA Code-R (Intelligent Systems)

DEVELOPMENT TEAM: MIT, JSC

Milestones	FY02	FY03	FY04
Hybrid Mode Estimation	X		
Learning as Hybrid Expectation Maximization		X	
Demonstration on Bioplex and Mars EDL		X	X
•Decomposition algorithms			X

NASA RELEVANCE:

Recent mission failures (e.g., Mars Climate Orbiter and Polar Lander) highlight the need for monitoring capabilities that detect subtle symptoms, and simulators that can be quickly tailored to a mission. Our approach enables:

- predictive diagnosis and the detection of incipient failures that are hidden within noise.
- generation of estimators that track system state across changes in system modes.
- prototyping of simulators that acquire their physical models automatically.